KNOWLEDGE GRADIENT METHODS FOR BAYESIAN OPTIMIZATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jian Wu

August 2017

KNOWLEDGE GRADIENT METHODS

FOR BAYESIAN OPTIMIZATION

Jian Wu, Ph.D.

Cornell University 2017

Bayesian optimization, a framework for global optimization of expensive-to-evaluate functions, has shown success in machine learning and experimental design because it is able to find global optima with a remarkably small number of potentially noisy objective function evaluations. In this dissertation, we study in detail how the concept of the knowledge gradient (KG) can be adopted to design novel Bayesian optimization algorithms.

First, we propose a novel parallel Bayesian optimization algorithm by generalizing the concept of KG from the fully sequential setting to the parallel setting (qKG). By construction, this method provides a one-step Bayes-optimal batch of points to sample. We provide an efficient strategy for computing this Bayes-optimal batch of points, and we demonstrate that the parallel knowledge gradient method finds global optima significantly faster than previous batch Bayesian optimization algorithms on both synthetic test functions and when tuning hyperparameters of practical machine learning algorithms, especially when function evaluations are noisy.

Second, we present a novel discretization-free strategy to calculate the set of points to be evaluated under the knowledge gradient method when used over an continuous domain. KG methods are widely studied for discrete ranking and selection problems, and provide a one-step Bayes-optimal point to sample, but all the previous efforts generalizing KG to continuous domains rely on a discretized

finite approximation due to the computational challenges in calculating KG. However, the discretization introduces error, and scales poorly as the dimension of the domain grows. In this chapter, we develop a fast discretization-free knowledge gradient method for Bayesian optimization, which is useful for all settings where KG is used over an continuous domain that overcomes these challenges.

Third, we explore the "what, when, and why" of Bayesian optimization with derivative information. We also develop a Bayesian optimization algorithm that effectively leverages gradients. This algorithm accommodates incomplete and noisy gradient observations, can be used in both the sequential and batch settings, and can optionally reduce the computational overhead of inference by selecting the single most valuable directional derivative to retain. For this purpose, we develop a novel acquisition function, called the derivative-enabled knowledge-gradient (dKG). This generalizes the previously proposed batch knowledge gradient method to the derivative setting. We also provide a theoretical analysis of the algorithm: it is one-step Bayes-optimal by construction when derivatives are available, and we show (1) that it provides one-step value greater than in the derivative-free setting; and (2) that its estimator of the global optimum is asymptotically consistent.

Fourth, we show some preliminary results on how KG can be adopted to settings where we have some low-fidelity but cheap approximations. To this end, we develop a novel Bayesian optimization algorithm, continuous-fidelity knowledge gradient (cfKG), which can adaptively choose both the fidelity and the desired point to sample by better balancing the trade-off between the information gain vs. the cost when we have some continuous parameters controlling the fidelity of the information source we can query. Some preliminary numerical results are shown.

BIOGRAPHICAL SKETCH

Jian Wu was born in Hefei, Anhui province in China, a beautiful city in the middle of China. Before his college, he spent most of his time at his hometown. From age sixteen, he began his wonderful college life at Tsinghua, where he received a B.Eng in Automation. During his stay in Tsinghua, he obtained a foundation in math and computer science, which provides many building blocks for the work in this dissertation. Since joining Cornell ORIE, he has been interested in many practical problems arising in modern society including materials design [86], queueing theory [7, 8] and Bayesian optimization [84, 87].

To my wife, Linran.

Her love, trust, and support makes this work possible!

ACKNOWLEDGEMENTS

I am grateful to many those who have contributed towards completing this dissertation.

First and foremost, I would like to thank my advisors, Professor Peter I. Frazier and Jim Dai, for their constant help throughout my stay at Cornell. Their ways of defining and thinking about problems inspire me a lot. I am also grateful for their encouragements when sometimes I decided to pursue my own ideas. Professor Frazier's willingness to apply Operations Research to solve problems from very diversified fields teaches me how useful OR can be. I also thank Professor Frazier, for his financial support for attending academic conferences, to providing computational resources.

I am also grateful to my family. Especially, I would like to thank my wife, Linran, for her love and trust. This work will not be possible without her. I dedicate this thesis to her.

Finally, I would like to thank my school of Operations Research & Information Engineering at Cornell, for the super productive and collaborative environment.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . vi
List of Figures . . . viii

1 Introduction **1**
1.1 Examples Problems . . . 3
1.2 Bayesian Optimization . . . 5
1.2.1 Gaussian Processes . . . 6
1.2.2 Acquisition Functions . . . 8
1.3 Thesis Organization . . . 14

2 The Parallel Knowledge Gradient Method for Batch Bayesian Optimization **18**
2.1 Introduction . . . 18
2.2 Related work . . . 20
2.3 Background on Gaussian processes . . . 22
2.4 Parallel knowledge gradient (q-KG) . . . 23
2.5 Computation of q-KG . . . 25
2.5.1 Estimating q-KG when $\mathbb{A}$ is finite in (2.4.1) . . . 26
2.5.2 Estimating the gradient of q-KG when $\mathbb{A}$ is finite in (2.4.1) . . . 26
2.5.3 Approximating q-KG when $\mathbb{A}$ is infinite in (2.4.1) through discretization . . . 27
2.6 Numerical experiments . . . 28
2.6.1 Noise-free problems . . . 29
2.6.2 Noisy problems . . . 32
2.7 Summary . . . 34

3 Discretization-free Knowledge Gradient Methods for Bayesian Optimization **36**
3.1 Introduction . . . 36
3.2 Discretization-free computation of q-KG . . . 36
3.2.1 Estimating q-KG . . . 37
3.2.2 Estimating the gradient of q-KG . . . 38
3.2.3 Bayesian Treatment of Hyperparameters. . . . 39
3.2.4 Asynchronous q-KG Optimization . . . 40
3.3 Proof of Theorem 1 . . . 41
3.4 Summary . . . 43

4 Bayesian Optimization with Gradients **44**
4.1 Introduction . . . 44
4.2 Related Work . . . 46
4.3 Knowledge Gradient with Derivatives . . . 48
4.3.1 Derivative Information . . . 48
4.3.2 The *d-KG* Acquisition Function . . . 49
4.3.3 Efficient Exact Computation of *d-KG* . . . 53
4.3.4 Theoretical Analysis . . . 55
4.4 Experiments . . . 56
4.4.1 Synthetic Test Functions . . . 57
4.4.2 Real-World Test Functions . . . 57
4.5 Summary . . . 61

5 Continuous-Fidelity Knowledge Gradient for Fast Bayesian Optimization **63**
5.1 Introduction . . . 63
5.2 Continuous-Fidelity Knowledge Gradient . . . 64
5.3 Summary . . . 66

6 Conclusion **68**

A Supplementary Materials for Chapter 2 **71**
A.1 Asynchronous *q-KG* Optimization . . . 71
A.2 Speed-up analysis . . . 71
A.3 The unbiasedness of the stochastic gradient estimator . . . 72
A.3.1 The proof of condition (i) . . . 73
A.3.2 The proof of condition (ii) . . . 74
A.3.3 The proof of condition (iii) . . . 75
A.4 The convergence of stochastic gradient ascent . . . 76

B Supplementary Materials for Chapter 4 **77**
B.1 The Posterior Distribution of the Multi-Output GP . . . 77
B.2 The Computation of *d-KG* and its Gradient: Additional Details . . . 77
B.3 Additional Experimental Results . . . 79
B.4 Proof of Proposition 3 and Proposition 4 . . . 81
B.5 Proof of Theorem 2 . . . 83

LIST OF FIGURES

1.1 Illustration of Gaussian process regression with noisy evaluations on a 1-d function. The dots show previously evaluated points, $(x^{(i)}, f(x^{(i)}))$. The solid line shows the posterior mean, $\mu^{(n)}(x)$ as a function of x, which is an estimate $f(x)$, and the shadow area show a Bayesian confidence interval for each $f(x)$, calculated as $\mu^{(n)}(x) \pm 1.96\sigma^{(n)}(x)$. 7

1.2 The source is at https://people.orie.cornell.edu/pfrazier/presentations.html. One should note that we are solving the maximization problem in this figure (instead of the minimization problem in (1.2.1)).The Upper panel shows the posterior distribution in a randomly sampled problem with independent normal homoscedastic noise and a one-dimensional input space, where the black solid line is the true function, the circles are previously measured points, the red solid line is the posterior mean $\mu^{(n)}(x)$, and the red dashed lines are at $\mu^{(n)}(x) \pm 1.96\sigma^{(n)}(x)$. Lower panel shows the natural logarithm of the knowledge gradient factor $\texttt{KG}(x)$ computed from this posterior distribution. An "x" is marked at the point with the largest KG factor, which is where the KG algorithm would evaluate next. 13

2.1 Performances on noise-free synthetic functions with $q = 4$. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of function evaluations. 30

2.2 Performances on tuning machine learning algorithms with $q = 4$ 31

2.3 Performances on noisy synthetic functions with $q = 4$. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of function evaluations. 33

2.4 Tuning logistic regression on smaller test sets with $q = 4$ 34

4.1 The topmost plots show (1) the posterior surfaces of a function sampled from a one dimensional GP with and without incorporating observations of the gradients. Note that the posterior variance is smaller if the gradients are incorporated; (2) the utility of sampling each point under the value of information criteria of KG and EI in both settings. If no derivatives are observed, both KG and EI will query a point with high potential gain (i.e. a small expected function value). On the other hand, when gradients are observed, *d-KG* makes a considerably better sampling decision, whereas *d-EI* samples essentially the same location as *EI*. The plots in the bottom row depict the posterior surface *after* the respective sample. Interestingly, KG benefits more from observing the gradients than EI (the last two plots): *d-KG* samples a point whose observation yields an accurate knowledge of the location of the optimum, while *d-EI* still has considerable uncertainty around the optimum. 52

4.2 The average performance of 100 replications (the log10 of the immediate regret vs. the number of function evaluations). *d-KG* performs significantly better than its competitors for all benchmarks. In Branin and Hartmann, we also plot black lines, which is the performance of BFGS. . 58
4.3 Results for the weighted KNN benchmark, the spectral mixture kernel benchmark, logistic regression and deep neural network (from left to right), all with batch size 8 and averaged over 20 replications. 61

5.1 The comparisons between KG and continuous-fidelity KG on three synthetic functions: Branin, 3-d Hartmann, and Rosenbrock. The fidelity space is one dimensional for Branin and Hartmann3 and two dimensional for Rosenbrock. 66

A.1 The performances of *q-KG* with different batch sizes. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of iterations. Iteration 0 is the initial designs. For each iteration later, we evaluate q points recommended by the *q-KG* algorithm. 72

B.1 The average performance of 100 replications (the log10 of the immediate regret vs. the number of function evaluations) for the Levy and Ackley functions. For the Ackley function, we assume that a noisy observation of the full gradient is available. On the Levy function only 4th partial derivative can be observed (with noise). *d-KG* performs significantly better than its competitors for all benchmarks except the Levy function. 81

CHAPTER 1

INTRODUCTION

This thesis considers global optimization of expensive functions, in which (1) our objective function is expensive to evaluate (in terms of resources, money, time, etc), such that the number of function evaluations we can perform is extremely limited; (2) evaluating the objective function provides the noisy value of the objective, and possibly some partial derivatives obscured by noise; (3) the objective function lacks structure beyond continuity such as convexity or submodularity; (4) we seek a global, rather than a local, optimum. Such problems typically arise when the objective function is evaluated by running a complex computer code (e.g. materials design [15], hyperparameter tuning [66], or algorithm configuration [26]), but also arises when the objective function can only be evaluated by performing a labor-intensive experiment, or running A-B tests in the real world.

Bayesian optimization (BO) methods are one class of methods attempting to solve such problems, and contain two components: (1) a statistical model: they use machine learning to build a statistical model for the unknown objective function, and the model provides not only point predictions but uncertainty quantifications; (2) an acquisition function: they use a carefully designed acquisition function to suggest which point(s) in the function's domain would be the most valuable to evaluate next. The acquisition function should balance the trade-off between exploration and exploitation.

The most common statistical model in BO is to put a Gaussian process [61] prior distribution on the function f, updating this prior distribution with each new observation of f. Some other models such as random forest [26] or deep neural network [67, 70] have also been explored. Common acquisition functions choos-

ing the next point or points to evaluate include probability of improvement (PI) [74], expected improvement (EI) [30], upper confidence bound (UCB) [71], entropy search/predictive entropy search (ES/PES) [22, 23], and knowledge gradient (KG) [62]. By carefully measuring the benefit of sampling at a point, BO often finds "near optimal" function values with fewer evaluations in comparison with other global optimization algorithms [66].

To the best of our knowledge, BO was pioneered in [37] followed by several seminal papers in the 1970s and 1980s pursued in [53] and [51]. In the late 1990s, Jones, Schonlau, and Welch introduced the famous Efficient Global Optimization (EGO) method [30], building on the previous work by Mockus [51]. It used the Gaussian process (GP) as its statistical model and employed the EI acquisition function. This method became very popular and well-known in engineering, where it has been adopted for many time-consuming engineering design applications. In the 2000s, many follow-up works in statistics and engineering [24, 46] continued the successful story of BO. Since 2010s, interest in BO rose in the machine learning community, with publications including [26, 66], after it was discovered BO was an excellent tool for tuning hyperparameters of computationally expensive machine learning models. If one is interested in knowing more about Bayesian optimization, [4] serves as a basic tutorial article and [64] gives a comprehensive review of some ongoing developments in BO.

In this chapter, we first enumerate a few problems arising from engineering design, materials discovery, medical science and machine learning that are suitable for Bayesian optimization in Section 1.1. We then present the mathematical formulation of problems considered by Bayesian optimization in Section 1.2, and introduce the most common statistical model used by Bayesian optimization, i.e.

the Gaussian Process (GP), in Section 1.2.1, and common acquisition functions (the criteria for suggesting the point(s) to evaluate next) in Section 1.2.2. Finally, we provide an overview for the rest of the thesis in Section 1.3.

1.1 Examples Problems

The problems that can be solved by Bayesian optimization are incredibly common in practice. We enumerate a few examples from a long list, arising from a diversified set of fields. Some of these examples (such as machine learning hyperparameter tuning) will be considered more fully in the later chapters, and others are included to underscore the broad applicability of Bayesian optimization.

- **Materials design and discovery:** We would like to choose the chemical structure, composition, or processing conditions of a material to meet some design criteria. The evaluation usually requires physical experiments which are both time-consuming and expensive. Therefore, we choose the design adaptively by BO and test how good it is. See [15].
- **Drug discovery:** We would like to choose a molecule among a large pool to find the one that best treats a given disease. Synthesizing and testing a compound may require days and a significant investment of lab materials, which strictly limits the number of tests. Therefore, we test the molecules adaptively by BO, and then collect information about its effectiveness. See [55].
- **Algorithm configuration:** We would like to choose the algorithm parameters of some start-of-art solvers for hard computational problems. Each configuration evaluation is time-consuming and takes a lot of computational resources. Therefore, we test configurations adaptively by BO and collect

the information about their empirical performance on standard benchmarks. See [26].

- **Adaptive MCMC:** We would like to choose the parameters in the Hybrid Monte Carlo (HMC) algorithm, such as step size, to ensure mixing of the Markov chain. Each parameter evaluation requires running HMC on the whole training data and is time-consuming. Therefore, we choose the parameters adaptively by BO. See [47].
- **Reinforcement learning:** We would like to choose a parametric policy in a challenging Reinforcement Learning (RL) application. BO is attractive for this problem because it exploits Bayesian prior information about the expected return and exploits this knowledge to select new policies to execute. Effectively, the BO framework for policy search addresses the exploration-exploitation tradeoff. See [44].
- **Machine learning hyperparameter tuning:** For some machine learning algorithms, e.g., deep neural networks, using high-quality hyperparameters instead of low-quality ones is the difference between state-of-the-art predictive performance and being essentially useless. Typical approaches to tuning hyperparameters include hand tuning by experts and brute-force search. However, as the number of parameters grow, these approaches quickly become infeasible. To overcome this challenge, BO can be used to automate hyperparameter tunning. See [66].

1.2 Bayesian Optimization

In conventional Bayesian optimization (BO) [30], we wish to optimize a derivative-free expensive-to-evaluate function f with feasible domain $\mathbb{A} \subseteq \mathbb{R}^d$,

$$\min_{\boldsymbol{x} \in \mathbb{A}} f(\boldsymbol{x}), \tag{1.2.1}$$

with as few function evaluations as possible. In this thesis, we assume that membership in the domain $\mathbb{A}$ is easy to evaluate and we can evaluate f only at points in $\mathbb{A}$. We assume that evaluations of f are either noise-free, or have additive independent normally distributed noise.

In many real problems, we do not know any structure information of the objective function such as convexity or submodularity, and can only hope to obtain an estimate of the function value through running simulations or conducting physical experiments. Derivative information about the objective functions is not available in most of cases (that is why BO often belongs to the class of derivative-free optimization methods). Moreover, estimating $f(\boldsymbol{x})$ is usually an expensive process, for example, training a complex machine learning model such as denseNet [25] on a huge dataset such as ImageNet takes several days; conducting physical experiments requires labor investment of researchers and consumes resources.

Bayesian optimization is particularly suitable for these problems, when the objective function does not have an explicit form, derivative information is not readily available, and function evaluation is expensive. BO consists of two components: a statistical model and an acquisition function. We will go over one by one: first, Bayesian optimization can incorporate the expertise belief about the problem in the form of a Bayesian prior; second, the sampling approach is designed to balance the trade-off between exploration vs. exploitation, reducing the number of function

evaluations required to find the optimum. We will review both aspects in details in Section 1.2.1 and Section 1.2.2.

1.2.1 Gaussian Processes

Since the objective function is assumed unknown, in Bayesian optimization, we first build a predictive model on the objective function using Bayesian statistics. Among the wide variety of Bayesian statistical methods, Gaussian Process (GP) regression is a popular choice in Bayesian optimization. A Gaussian Process is a probability distribution over functions. Under the Gaussian Process, the marginal probability distribution of the value of the function at any single point is a normal distribution. The joint distribution of the values of the function at any collection of points is a multivariate normal distribution.

In Gaussian process regression, we use a Gaussian Process as our prior probability distribution over the unknown objective function. We put a Gaussian process prior over the function $f : \mathbb{A} \to \mathbb{R}$, which is specified by its mean function $\mu(\boldsymbol{x}) : \mathbb{A} \to \mathbb{R}$ and kernel function $K(\boldsymbol{x_1}, \boldsymbol{x_2}) : \mathbb{A} \times \mathbb{A} \to \mathbb{R}$. We assume either exact or independent normally distributed measurement errors, i.e. the evaluation $\boldsymbol{y}(\boldsymbol{x}_i)$ at point $\boldsymbol{x}_i$ satisfies

$$\boldsymbol{y}(\boldsymbol{x}_i) \mid f(\boldsymbol{x}_i) \sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma^2(\boldsymbol{x}_i)),$$

where $\sigma^2 : \mathbb{A} \to \mathbb{R}^+$ is a known function describing the variance of the measurement errors. If σ^2 is not known, we can also estimate it by either *maximum likelihood estimation* (MLE) or in the fully Bayesian way.

Supposing we have measured f at n points $\boldsymbol{x}^{(1:n)} := \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}\}$ and

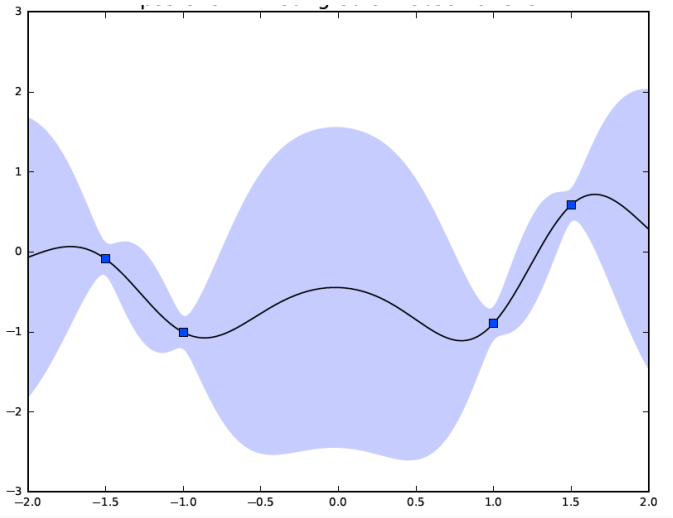

Figure 1.1: Illustration of Gaussian process regression with noisy evaluations on a 1-d function. The dots show previously evaluated points, $(x^{(i)}, f(x^{(i)}))$. The solid line shows the posterior mean, $\mu^{(n)}(x)$ as a function of x, which is an estimate $f(x)$, and the shadow area show a Bayesian confidence interval for each $f(x)$, calculated as $\mu^{(n)}(x) \pm 1.96\sigma^{(n)}(x)$.

obtained corresponding measurements $y^{(1:n)}$, we can then combine these observed function values with our prior to obtain a posterior distribution on f. This posterior distribution is still a Gaussian process with the mean function $\boldsymbol{\mu}^{(n)}$ and the kernel function $K^{(n)}$ as follows

$$\begin{aligned}
\boldsymbol{\mu}^{(n)}(\boldsymbol{x}) &= \mu(\boldsymbol{x}) + K(\boldsymbol{x}, \boldsymbol{x}^{(1:n)}) \\
&\quad \left(K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)}) + \mathrm{diag}\{\sigma^2(\boldsymbol{x}^{(1)}), \cdots, \sigma^2(\boldsymbol{x}^{(n)})\}\right)^{-1} (y^{(1:n)} - \mu(\boldsymbol{x}^{(1:n)})), \\
K^{(n)}(\boldsymbol{x_1}, \boldsymbol{x_2}) &= K(\boldsymbol{x_1}, \boldsymbol{x_2}) \\
&\quad - K(\boldsymbol{x_1}, \boldsymbol{x}^{(1:n)}) \left(K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)}) + \mathrm{diag}\{\sigma^2(\boldsymbol{x}^{(1)}), \cdots, \sigma^2(\boldsymbol{x}^{(n)})\}\right)^{-1} K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x_2}).
\end{aligned} \tag{1.2.2}$$

Figure 1.1 shows the output from Gaussian process regression on a one-dimensional function. [15] offers a comprehensive review of Gaussian process regression, including the choice of the mean function and the kernel function, inference with noisy observations, and hyperparameters estimation.

1.2.2 Acquisition Functions

The second crucial piece of Bayesian optimization is making good decisions about where to direct future sampling. Bayesian optimization methods address this by using a measure of the benefit that would be gained by sampling at a point, commonly known as an "acquisition function". Much of the literature in BO focuses on designing good acquisition functions that reach optima with as few evaluations as possible. Maximizing this acquisition function usually provides a single point to evaluate next, with common acquisition functions for sequential Bayesian optimization including probability of improvement (PI) [37], expected improvement (EI) [30, 52], upper confidence bound (UCB) [71], entropy search/predictive entropy search (ES/PES) [22, 23], and knowledge gradient (KG) [13, 62]. We will introduce some of these acquisition functions most relevant to this thesis in details.

Probability of Improvement

Considering the setting of noise-free function evaluations, the early work of [37] suggested maximizing the probability of improvement over the best sampled value observed so far, written as

$$\texttt{PI}(\boldsymbol{x}) = \mathbb{P}\left(f(\boldsymbol{x}) \leq f_n^* - \epsilon\right), \tag{1.2.3}$$

where $f_n^* = \min_{m \leq n} f(\boldsymbol{x}^{(m)})$, is the best sampled value at nth iteration, and ϵ is a positive constant that controls how much improvement over the current best sampled value is desired. Recall in (1.2.2) that if we have not observed $f(\boldsymbol{x})$ yet, $f(\boldsymbol{x})$ is a random variable that follows a normal distribution with mean $\mu^{(n)}(\boldsymbol{x})$, and standard deviation $\sigma^{(n)}(\boldsymbol{x})$. The superscript n here means the distribution is

a posterior after observing n data points. Then we can write (1.2.3) as

$$\texttt{PI}(\boldsymbol{x}) = \Phi\left(\frac{f_n^* - \epsilon - \mu^{(n)}(\boldsymbol{x})}{\sigma^{(n)}(\boldsymbol{x})}\right), \tag{1.2.4}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

The choice of ϵ is a tunable parameter, although [37] suggested that in general ϵ should start fairly high early in the optimization, to drive exploration, and decrease toward zero as the algorithm continued to engage more effort in exploitation at the end. Several works have studied the empirical impact of choices of ϵ [29, 43, 74].

Upper/Lower confidence bound

Dating back to the seminal work of [38] on the multi-armed bandit problem, the upper confidence bound criterion has been a popular way of negotiating exploration and exploitation, often with provable cumulative regret bounds. More recently, the Gaussian process upper confidence bound (GP-UCB [71]) algorithm was proposed as a Bayesian optimistic algorithm with provable cumulative regret bounds. In UCB (LCB for minimization problems), we wish to sample at a point x which minimizes the following quantile.

$$\text{LCB}(x) = \mu^n(x) - \beta_n \sigma^n(x).$$

Similar to ϵ above, tuning β_n can provide a performance boost.

Expected Improvement

A more widely-used as well as satisfying alternative acquisition function would not only consider the probability of improvement, but also the magnitude of this

improvement. [30] proposed such an alternative, called expected improvement (EI). As its name suggests, the formulation is

$$\mathrm{EI}(\boldsymbol{x}) = E_n\left[(f_n^* - f(\boldsymbol{x}))^+\right], \tag{1.2.5}$$

where $E_n[\cdot]$ is the conditional expectation given previous n evaluations. Since the $f(\boldsymbol{x})$ follows a Gaussian distribution, the expectation in (1.2.5) can be written more explicitly, in terms of the normal cumulative distribution function $\Phi(\cdot)$, and the normal probability density function $\varphi(\cdot)$:

$$\begin{aligned} \mathrm{EI}(\boldsymbol{x}) &= \left(f_n^* - \mu^{(n)}(\boldsymbol{x})\right)\Phi\left(\gamma(x)\right) + \sigma^{(n)}(\boldsymbol{x})\varphi\left(\gamma(x)\right) \\ \gamma(\boldsymbol{x}) &= \frac{f_n^* - \mu^{(n)}(\boldsymbol{x})}{\sigma^{(n)}(\boldsymbol{x})}. \end{aligned} \tag{1.2.6}$$

The first advantage of this formulation compared with PI and also UCB (LCB) is that, without a user defined controlling variable ϵ/β_n, the expected improvement balances the tradeoff between exploration and exploitation automatically. In fact, the expected improvement favors point that, on the one hand, have a large predicted value, while on the other hand, have a significant amount of uncertainty to allow room for improvement. The second advantage is that EI is designed to be one-step Bayes-optimal under two conditions: (1) the function evaluation is noise-free; (2) the final recommendation is restricted to a previously sampled point. More details are coming in the next section.

In many applications, e.g., those involving physical experiments or stochastic simulations, function evaluations are noisy. The formulation (1.2.5) is not applicable in this case, because (1) f_n^* is not well-defined under noisy observations, in practice, f_n^* will be replaced by $\min y^{1:n}$ but it will lose its Bayes-optimal motivation and (2) it does not account for the prediction uncertainty at the current best

point. To alleviate these difficulties, alternative formulations of expected improvement were proposed in literature, e.g. SKO in [24].

Knowledge Gradient

Knowledge gradient (KG) [13, 62] fully accounts for the introduction of noise, and does not restrict the final recommendation to a previously sampled point. makes it possible to explore a class of solutions broader than just those that have been previously evaluated when recommending the final solution.

The knowledge gradient policy in [13] for discrete $\mathbb{A}$ chooses the next sampling decision by maximizing the expected incremental value of a measurement, without assuming (as expected improvement does) that the point returned as the optimum must be a previously sampled point. In this section, we will show that how KG can be generalized to continuous domains.

Suppose that we have observed n function values. If we were to stop measuring now, $\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x)$ would be the minimum of the predictor of the GP. If instead we took one more sample, $\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n+1)}(x)$ would be the minimum of the predictor of the GP. The difference between these quantities, $\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n+1)}(x)$, is the increment in expected solution quality (given the posterior after $n+1$ samples) that results from the additional sample.

This increment in solution quality is random given the posterior after n samples, because $\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n+1)}(x)$ is itself a random vector due to its dependence on the outcome of the future sample. We can compute the probability distribution of this difference, and the KG algorithm values the sampling decision $x^{(n+1)} = z$ according to its expected value, which we call the knowledge gradient factor, and indicate it

using the notation KG. Formally, we define the KG factor for a candidate point to sample z as

$$\mathrm{KG}(z, \mathbb{A}) = \min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n+1)}(x) | x^{(n+1)} = z \right], \tag{1.2.7}$$

where $\mathbb{E}_n [\cdot] := \mathbb{E} \left[\cdot | \boldsymbol{x}^{(1:n)}, y^{(1:n)} \right]$ is the expectation taken with respect to the posterior distribution after n evaluations. Then we choose to evaluate the next point that maximizes the knowledge gradient factor,

$$\max_{z \in \mathbb{A}} \mathrm{KG}(z, \mathbb{A}). \tag{1.2.8}$$

By construction, the knowledge gradient policy is Bayes-optimal for minimizing the minimum of the predictor of the GP if only one decision is remaining. The KG algorithm will reduce to the EI algorithm if function evaluations are noise-free and the final recommendation is restricted to the previous sampling decisions. Because under the two conditions above, the increment in expected solution quality will become

$$\begin{aligned} \min_{x \in \boldsymbol{x}^{(1:n)}} \boldsymbol{\mu}^{(n)}(x) - \min_{x \in \boldsymbol{x}^{(1:n)} \cup \{z\}} \boldsymbol{\mu}^{(n+q)}(x) &= \min y^{(1:n)} - \min \left\{ y^{(1:n)}, \min_{x \in \boldsymbol{z}^{(1:q)}} \boldsymbol{\mu}^{(n+1)}(x) \right\} \\ &= \left(\min y^{(1:n)} - \min_{x \in \boldsymbol{z}^{(1:q)}} \boldsymbol{\mu}^{(n+1)}(x) \right)^+, \end{aligned}$$

which is exactly the EI acquisition function. Thus, the knowledge gradient algorithm generalizes the expected improvement algorithm.

The KG factor for a one-dimensional optimization problem with noise is depicted in Figure 1.2 (one can find the source at https://people.orie.cornell.edu/pfrazier/presentations.html). We see a clear tradeoff between exploration and exploitation, where the KG factor favors measuring points with a large $\mu^{(n)}(\boldsymbol{x})$ and/or a large $\sigma^{(n)}(\boldsymbol{x})$. We also see local minima of the KG locate at points where we previously evaluated, just as with the expected improvement, but because there is

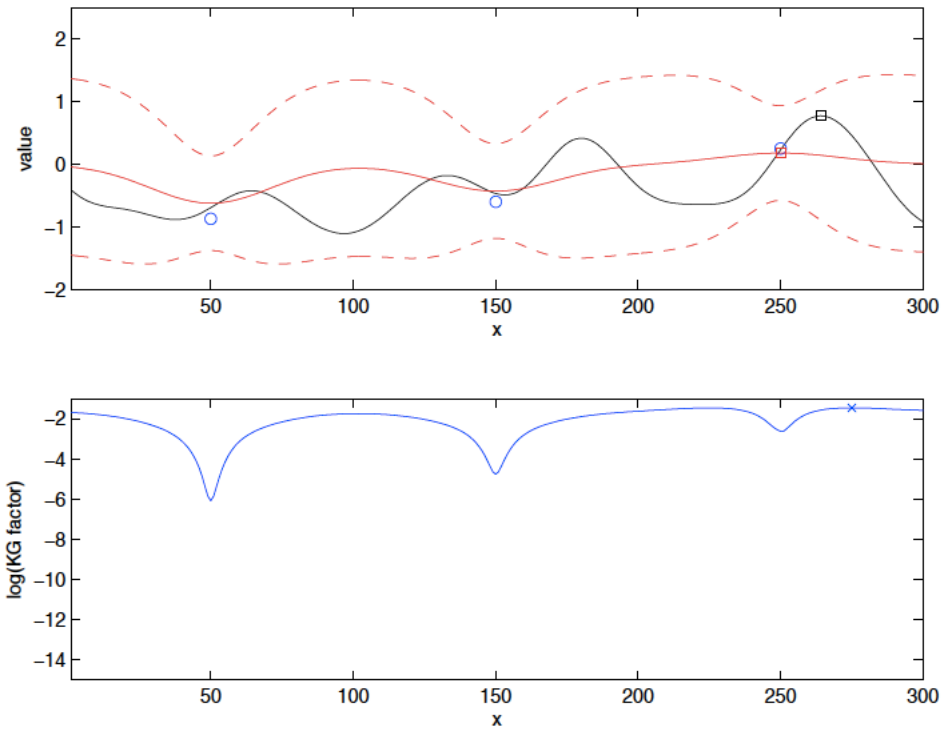

Figure 1.2: The source is at https://people.orie.cornell.edu/pfrazier/presentations.html. One should note that we are solving the maximization problem in this figure (instead of the minimization problem in (1.2.1)).The Upper panel shows the posterior distribution in a randomly sampled problem with independent normal homoscedastic noise and a one-dimensional input space, where the black solid line is the true function, the circles are previously measured points, the red solid line is the posterior mean $\mu^{(n)}(x)$, and the red dashed lines are at $\mu^{(n)}(x) \pm 1.96\sigma^{(n)}(x)$. Lower panel shows the natural logarithm of the knowledge gradient factor $\texttt{KG}(x)$ computed from this posterior distribution. An "x" is marked at the point with the largest KG factor, which is where the KG algorithm would evaluate next.

noise in our samples, the value at these points is not 0 — indeed, when there is noise, it may be useful to sample repeatedly at a point.

For tractability, one usually discretizes the continuous domain $\mathbb{A}$ to a finite approximation set $\bar{\mathbb{A}}$, then KG can approximated as

$$\overline{\mathrm{KG}}(z, \bar{\mathbb{A}}) = \min_{x \in \bar{\mathbb{A}}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \bar{\mathbb{A}}} \boldsymbol{\mu}^{(n+1)}(x) | z \right],$$

Still, calculating this expectation is more involved than calculating the expected improvement, but nevertheless can also be done analytically in terms of the normal

pdf and normal cdf. This is described in more details in [13].

The KG factor depends on the choice of the set $\bar{\mathbb{A}}$. Typically, to achieve a better result, we choose the set $\bar{\mathbb{A}}$ to contain more elements, allowing μ_n^* and μ_{n+1}^* to range over a representative portion of the space, and allowing the KG factor calculation to more accurately approximate the value that would result if we implemented the best option. However, the trade-off is that as we increase the size of $\bar{\mathbb{A}}$, computing the KG factor is much slower, making implementation of the KG method more computationally intensive.

1.3 Thesis Organization

This thesis is built upon the author's previously published works: Chapter 2 is based on [84]; Chapter 3 is the key contribution of a working paper by the author, which was submitted to 2017 Informs ICS student paper competition and Informs DM best paper competition and is in preparation for journal submission [85] at the time of completing this thesis; Chapter 4 is based on [87]. The code in this thesis is available at https://github.com/wujian16/qKG. We now give an overview of each chapter as follows:

Chapter 2

Chapter 2 considers batch Bayesian optimization. Chapter 2 proposes a novel batch BO method which measures the information gain of evaluating q points via a new acquisition function, the parallel knowledge gradient (q-KG). This method is derived using a decision-theoretic analysis that chooses the set of points to

evaluate next that is optimal in the average-case with respect to the posterior when there is only one batch of points remaining. Naively maximizing *q-KG* would be extremely computationally intensive, especially when q is large, and so, in this chapter, we develop a method based on infinitesimal perturbation analysis (IPA) [76] to evaluate *q-KG*'s gradient efficiently, allowing its efficient optimization. In our experiments on both synthetic functions and tuning practical machine learning algorithms, *q-KG* consistently finds better function values than other parallel BO algorithms, such as parallel EI [5, 66, 76], batch UCB [10] and parallel UCB with exploration [6]. *q-KG* provides especially large value when function evaluations are noisy.

Chapter 3

Chapter 3 presents a novel discretization-free strategy to provide the unbiased estimators of the KG acquisition function and its gradient when used over an continuous domain. KG methods are widely studied for discrete ranking and selection problems, which provide one-step Bayes-optimal point to sample. All the previous efforts generalizing KG to continuous domains rely on a discretized finite approximation due to the computational challenges in calculating KG. However, the discretization introduces error, and scales poorly as the dimension of domain grows. In this chapter, we develop a fast discretization-free knowledge gradient method for Bayesian optimization, which is useful for all settings where KG is used over an continuous domain.

Chapter 4

Chapter 4 considers Bayesian optimization with gradients. In this chapter, we explore the "what, when, and why" of Bayesian optimization with derivative information. We also develop a Bayesian optimization algorithm that effectively leverages gradients in various applications to outperform the state of the art. This algorithm accommodates incomplete and noisy gradient observations, can be used in both the sequential and batch settings, and can optionally reduce the computational overhead of inference by selecting the single most valuable directional derivatives to retain. For this purpose, we develop a new acquisition function, called the derivative-enabled knowledge-gradient (*d-KG*). This generalizes the previously proposed batch knowledge gradient method of [84] to the derivative setting, and replaces its approximate discretization-based method for calculating the knowledge-gradient acquisition function by a novel faster exact discretization-free method. We note that this discretization-free method is also of interest beyond the derivative setting, as it can be used to improve knowledge-gradient methods for other problem settings. We also provide a theoretical analysis of *d-KG* algorithm: it is one-step Bayes-optimal by construction when derivatives are available, and we show (1) that it provides one-step value greater than in the derivative-free setting; and (2) that its estimator of the global optimum is asymptotically consistent. In numerical experiments we compare with state-of-the-art batch Bayesian optimization algorithms with and without derivative information, and the gradient-based optimizer BFGS with full gradients.

Chapter 5

Chapter 5 shows some preliminary results on how KG can be adopted to settings where we have some low-fidelity but cheap approximations. To this end, we develop a novel Bayesian optimization algorithm, continuous-fidelity knowledge gradient (cfKG), which can adaptively choose both the fidelity and the desired point to sample by better balancing the trade-off between the information gain vs. the cost when we have some continuous parameters controlling the fidelity of the information source we can query. Some preliminary numerical results are shown.

Chapter 6

Chapter 6 summarizes the contributions of this thesis and describes some ongoing work in Bayesian optimization by this author: risk-averse knowledge gradient, scalable Bayesian optimization, and high dimensional Bayesian optimization.

CHAPTER 2

THE PARALLEL KNOWLEDGE GRADIENT METHOD FOR BATCH BAYESIAN OPTIMIZATION

2.1 Introduction

In Bayesian optimization [66] (BO), we wish to optimize a derivative-free expensive-to-evaluate function f with feasible domain $\mathbb{A} \subseteq \mathbb{R}^d$,

$$\min_{\boldsymbol{x} \in \mathbb{A}} f(\boldsymbol{x}),$$

with as few function evaluations as possible. In this chapter, we assume that membership in the domain $\mathbb{A}$ is easy to evaluate and we can evaluate f only at points in $\mathbb{A}$. We assume that evaluations of f are either noise-free, or have additive independent normally distributed noise. We consider the parallel setting, in which we perform more than one simultaneous evaluation of f.

BO typically puts a Gaussian process prior distribution on the function f, updating this prior distribution with each new observation of f, and choosing the next point or points to evaluate by maximizing an acquisition function that quantifies the benefit of evaluating the objective as a function of where it is evaluated. In comparison with other global optimization algorithms, BO often finds "near optimal" function values with fewer evaluations [66]. As a consequence, BO is useful when function evaluation is time-consuming, such as when training and testing complex machine learning algorithms (e.g. deep neural networks) or tuning algorithms on large-scale dataset (e.g. ImageNet) [9]. Recently, BO has become popular in machine learning as it is highly effective in tuning hyperparameters of machine learning algorithms [16, 17, 66, 72].

Most previous work in BO assumes that we evaluate the objective function sequentially [30], though a few recent papers have considered parallel evaluations [6, 10, 63, 76]. While in practice, we can often evaluate several different choices in parallel, such as multiple machines can simultaneously train the machine learning algorithm with different sets of hyperparameters. In this chapter, we assume that we can access $q \geq 1$ evaluations simultaneously at each iteration. Then we develop a new parallel acquisition function to guide where to evaluate next based on the decision-theoretical analysis.

Our Contributions. We propose a novel batch BO method which measures the information gain of evaluating q points via a new acquisition function, the parallel knowledge gradient (q-KG). This method is derived using a decision-theoretic analysis that chooses the set of points to evaluate next that is optimal in the average-case with respect to the posterior when there is only one batch of points remaining. Naively maximizing q-KG would be extremely computationally intensive, especially when q is large, and so, in this chapter, we develop a method based on infinitesimal perturbation analysis (IPA) [76] to evaluate q-KG's gradient efficiently, allowing its efficient optimization. In our experiments on both synthetic functions and tuning practical machine learning algorithms, q-KG consistently finds better function values than other parallel BO algorithms, such as parallel EI [5, 66, 76], batch UCB [10] and parallel UCB with exploration [6]. q-KG provides especially large value when function evaluations are noisy. The code in this chapter is available at https://github.com/wujian16/qKG.

The rest of the chapter is organized as follows. Section 2.2 reviews related work. Section 4.3.1 gives background on Gaussian processes and defines notation used later. Section 2.4 proposes our new acquisition function q-KG for batch BO.

Section 2.5 provides our computationally efficient approach to maximizing *q-KG*. Section 2.6 presents the empirical performance of *q-KG* and several benchmarks on synthetic functions and real problems. Finally, Section 2.7 concludes the chapter.

2.2 Related work

Within the past several years, the machine learning community has revisited BO [16, 17, 63, 66, 68, 72] due to its huge success in tuning hyperparameters of complex machine learning algorithms. BO algorithms consist of two components: a statistical model describing the function and an acquisition function guiding evaluations. In practice, Gaussian Process (GP) [61] is the mostly widely used statistical model due to its flexibility and tractability. Much of the literature in BO focuses on designing good acquisition functions that reach optima with as few evaluations as possible. Maximizing this acquisition function usually provides a single point to evaluate next, with common acquisition functions for sequential Bayesian optimization including probability of improvement (PI)[74], expected improvement (EI) [30], upper confidence bound (UCB) [71], entropy search (ES) [23], and knowledge gradient (KG) [62].

Recently, a few papers have extended BO to the parallel setting, aiming to choose a batch of points to evaluate next in each iteration, rather than just a single point. [18, 66] suggests parallelizing EI by iteratively constructing a batch, in each iteration adding the point with maximal single-evaluation EI averaged over the posterior distribution of previously selected points. [18] also proposes an algorithm called “constant liar", which iteratively constructs a batch of points to sample by maximizing single-evaluation while pretending that points previously

added to the batch have already returned values.

There are also work extending UCB to the parallel setting. [10] proposes the GP-BUCB policy, which selects points sequentially by a UCB criterion until filling the batch. Each time one point is selected, the algorithm updates the kernel function while keeping the mean function fixed. [6] proposes an algorithm combining UCB with pure exploration, called GP-UCB-PE. In this algorithm, the first point is selected according to a UCB criterion; then the remaining points are selected to encourage the diversity of the batch. These two algorithms extending UCB do not require Monte Carlo sampling, making them fast and scalable. However, UCB criteria are usually designed to minimize cumulative regret rather than immediate regret, causing these methods to underperform in BO, where we wish to minimize simple regret.

The parallel methods above construct the batch of points in an iterative greedy fashion, optimizing some single-evaluation acquisition function while holding the other points in the batch fixed. The acquisition function we propose considers the batch of points collectively, and we choose the batch to jointly optimize this acquisition function. Other recent papers that value points collectively include [5] which optimizes the parallel EI by a closed-form formula, [48, 76], in which gradient-based methods are proposed to jointly optimize a parallel EI criterion, and [63], which proposes a parallel version of the ES algorithm and uses Monte Carlo Sampling to optimize the parallel ES acquisition function.

We compare against methods from a number of these previous papers in our numerical experiments, and demonstrate that we provide an improvement, especially in problems with noisy evaluations.

Our method is also closely related to the knowledge gradient (KG) method [13, 62] for the non-batch (sequential) setting, which chooses the Bayes-optimal point to evaluate if only one iteration is left [62], and the final solution that we choose is not restricted to be one of the points we evaluate. (Expected improvement is Bayes-optimal if the solution is restricted to be one of the points we evaluate.) We go beyond this previous work in two aspects. First, we generalize to the parallel setting. Second, while the sequential setting allows evaluating the KG acquisition function exactly, evaluation requires Monte Carlo in the parallel setting, and so we develop more sophisticated computational techniques to optimize our acquisition function. Recently, [77] studies a nested batch knowledge gradient policy. However, they optimize over a finite discrete feasible set, where the gradient of KG does not exist. As a result, their computation of KG is much less efficient than ours. Moreover, they focus on a nesting structure from materials science not present in our setting.

2.3 Background on Gaussian processes

In this section, we state our prior on f, briefly discuss well known results about Gaussian processes (GP), and introduce notation used later. We put a Gaussian process prior over the function $f : \mathbb{A} \to \mathbb{R}$, which is specified by its mean function $\mu(\boldsymbol{x}) : \mathbb{A} \to \mathbb{R}$ and kernel function $K(\boldsymbol{x_1}, \boldsymbol{x_2}) : \mathbb{A} \times \mathbb{A} \to \mathbb{R}$. We assume either exact or independent normally distributed measurement errors, i.e. the evaluation $\boldsymbol{y}(\boldsymbol{x}_i)$ at point $\boldsymbol{x}_i$ satisfies

$$\boldsymbol{y}(\boldsymbol{x}_i) \mid f(\boldsymbol{x}_i) \sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma^2(\boldsymbol{x}_i)),$$

where $\sigma^2 : \mathbb{A} \to \mathbb{R}^+$ is a known function describing the variance of the measurement errors. If σ^2 is not known, we can also estimate it as we do in Section 2.6.

Supposing we have measured f at n points $\boldsymbol{x}^{(1:n)} := \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}\}$ and obtained corresponding measurements $y^{(1:n)}$, we can then combine these observed function values with our prior to obtain a posterior distribution on f. This posterior distribution is still a Gaussian process with the mean function $\boldsymbol{\mu}^{(n)}$ and the kernel function $K^{(n)}$ as follows

$$\begin{aligned}
\boldsymbol{\mu}^{(n)}(\boldsymbol{x}) &= \mu(\boldsymbol{x}) \\
&\quad + K(\boldsymbol{x}, \boldsymbol{x}^{(1:n)}) \left(K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)}) + \operatorname{diag}\{\sigma^2(\boldsymbol{x}^{(1)}), \cdots, \sigma^2(\boldsymbol{x}^{(n)})\}\right)^{-1} (y^{(1:n)} - \mu(\boldsymbol{x}^{(1:n)})), \\
K^{(n)}(\boldsymbol{x_1}, \boldsymbol{x_2}) &= K(\boldsymbol{x_1}, \boldsymbol{x_2}) \\
&\quad - K(\boldsymbol{x_1}, \boldsymbol{x}^{(1:n)}) \left(K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x}^{(1:n)}) + \operatorname{diag}\{\sigma^2(\boldsymbol{x}^{(1)}), \cdots, \sigma^2(\boldsymbol{x}^{(n)})\}\right)^{-1} K(\boldsymbol{x}^{(1:n)}, \boldsymbol{x_2}).
\end{aligned} \tag{2.3.1}$$

2.4 Parallel knowledge gradient (q-KG)

In this section, we propose a novel parallel Bayesian optimization algorithm by generalizing the concept of the knowledge gradient from [13] to the parallel setting. The knowledge gradient policy in [13] for discrete $\mathbb{A}$ chooses the next sampling decision by maximizing the expected incremental value of a measurement, without assuming (as expected improvement does) that the point returned as the optimum must be a previously sampled point.

We now show how to compute this expected incremental value of an additional iteration in the parallel setting. Suppose that we have observed n function values. If we were to stop measuring now, $\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n)}(x)$ would be the minimum of the predictor of the GP. If instead we took one more batch of samples, $\min_{x \in \mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x)$

would be the minimum of the predictor of the GP. The difference between these quantities, $\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x)$, is the increment in expected solution quality (given the posterior after $n+q$ samples) that results from the additional batch of samples.

This increment in solution quality is random given the posterior after n samples, because $\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x)$ is itself a random vector due to its dependence on the outcome of the samples. We can compute the probability distribution of this difference (with more details given below), and the *q-KG* algorithm values the sampling decision $\boldsymbol{z}^{(1:q)} := \{z_1, z_2, \cdots, z_q\}$ according to its expected value, which we call the parallel knowledge gradient factor, and indicate it using the notation *q-KG*. Formally, we define the *q-KG* factor for a set of candidate points to sample $\boldsymbol{z}^{(1:q)}$ as

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n+q)}(x) | \boldsymbol{z}^{(1:q)} \right], \tag{2.4.1}$$

where $\mathbb{E}_n[\cdot] := \mathbb{E}\left[\cdot | \boldsymbol{x}^{(1:n)}, y^{(1:n)}\right]$ is the expectation taken with respect to the posterior distribution after n evaluations. Then we choose to evaluate the next batch of q points that maximizes the parallel knowledge gradient,

$$\max_{\boldsymbol{z}^{(1:q)}\subset\mathbb{A}} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}). \tag{2.4.2}$$

By construction, the parallel knowledge gradient policy is Bayes-optimal for minimizing the minimum of the predictor of the GP if only one decision is remaining. The *q-KG* algorithm will reduce to the parallel EI algorithm if function evaluations are noise-free and the final recommendation is restricted to the previous sampling decisions. Because under the two conditions above, the increment in

expected solution quality will become

$$\begin{aligned}\min_{x\in\boldsymbol{x}^{(1:n)}} \boldsymbol{\mu}^{(n)}(x) - \min_{x\in\boldsymbol{x}^{(1:n)}\cup\boldsymbol{z}^{(1:q)}} \boldsymbol{\mu}^{(n+q)}(x) &= \min y^{(1:n)} - \min\left\{y^{(1:n)}, \min_{x\in\boldsymbol{z}^{(1:q)}} \boldsymbol{\mu}^{(n+q)}(x)\right\} \\ &= \left(\min y^{(1:n)} - \min_{x\in\boldsymbol{z}^{(1:q)}} \boldsymbol{\mu}^{(n+q)}(x)\right)^+,\end{aligned}$$

which is exactly the parallel EI acquisition function. However, computing *q-KG* and its gradient is very expensive. We will address the computational issues in Section 2.5. The full description of the *q-KG* algorithm is summarized as follows.

Algorithm 1 The *q-KG* algorithm

Require: the number of initial stage samples I, and the number of main stage sampling iterations N.

1: Initial Stage: draw I initial samples from a latin hypercube design in $\mathbb{A}$, $\boldsymbol{x}^{(i)}$ for $i = 1, \ldots, I$.
2: Main Stange:
3: **for** $s = 1$ to N **do**
4: Solve (2.4.2), i.e. get $(z_1^*, z_2^*, \cdots, z_q^*) = \text{argmax}_{\boldsymbol{z}^{(1:q)}\subset\mathbb{A}}\, q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A})$
5: Sample these points $(z_1^*, z_2^*, \cdots, z_q^*)$, re-train the hyperparameters of the GP by MLE, and update the posterior distribution of f.
6: **end for**
7: **return** $x^* = \text{argmin}_{x\in\mathbb{A}}\mu^{(I+Nq)}(x)$.

2.5 Computation of *q-KG*

In this section, we provide the strategy to maximize *q-KG* by a gradient-based optimizer. In Section 2.5.1 and Section 2.5.2, we describe how to compute *q-KG* and its gradient when $\mathbb{A}$ is finite in (2.4.1). Section 2.5.3 describes an effective way to discretize $\mathbb{A}$ in (2.4.1). The readers should note that there are two $\mathbb{A}$s here, one is in (2.4.1) which is used to compute the *q-KG* factor given a sampling decision $\boldsymbol{z}^{(1:q)}$. The other is the feasible domain in (2.4.2) ($\boldsymbol{z}^{(1:q)} \subset \mathbb{A}$) that we optimize over. We are discretizing the first $\mathbb{A}$.

2.5.1 Estimating q-KG when $\mathbb{A}$ is finite in (2.4.1)

Following [13], we express $\boldsymbol{\mu}^{(n+q)}(x)$ as

$$\begin{aligned}\boldsymbol{\mu}^{(n+q)}(x) &= \boldsymbol{\mu}^{(n)}(x) + K^{(n)}(x, \boldsymbol{z}^{(1:q)}) \left(K^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)})\right. \\ &\quad \left. + \text{diag}\{\sigma^2(\boldsymbol{z}^{(1)}), \cdots, \sigma^2(\boldsymbol{z}^{(q)})\}\right)^{-1} \left(y(\boldsymbol{z}^{(1:q)}) - \boldsymbol{\mu}^{(n)}(\boldsymbol{z}^{(1:q)})\right).\end{aligned}$$

Because $y(\boldsymbol{z}^{(1:q)}) - \boldsymbol{\mu}^{(n)}(\boldsymbol{z}^{(1:q)})$ is normally distributed with zero mean and covariance matrix $K^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) + \text{diag}\{\sigma^2(\boldsymbol{z}^{(1)}), \cdots, \sigma^2(\boldsymbol{z}^{(q)})\}$ with respect to the posterior after n observations, we can rewrite $\boldsymbol{\mu}^{(n+q)}(x)$ as

$$\boldsymbol{\mu}^{(n+q)}(x) = \boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) Z_q, \tag{2.5.1}$$

where Z_q is a standard q-dimensional normal random vector, and

$$\tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) = K^{(n)}(x, \boldsymbol{z}^{(1:q)})(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1},$$

where $D^{(n)}(\boldsymbol{z}^{(1:q)})$ is the Cholesky factor of the covariance matrix $K^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) + \text{diag}\{\sigma^2(\boldsymbol{z}^{(1)}), \cdots, \sigma^2(\boldsymbol{z}^{(q)})\}$. Now we can compute the q-KG factor using Monte Carlo sampling when $\mathbb{A}$ is finite: we can sample Z_q, compute (2.5.1), then plug in (2.4.1), repeat many times and take average.

2.5.2 Estimating the gradient of q-KG when $\mathbb{A}$ is finite in (2.4.1)

In this section, we propose an unbiased estimator of the gradient of q-KG using IPA when $\mathbb{A}$ is finite. Accessing a stochastic gradient makes optimization much easier. By (2.5.1), we express q-KG as

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \mathbb{E}_{Z_q}\left(g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q)\right), \tag{2.5.2}$$

where $g = \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)})Z_q\right)$. Under the condition that μ and K are continuously differentiable, one can show that (please see the details in the supplementary materials)

$$\frac{\partial}{\partial z_{ij}} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) \quad = \quad \mathbb{E}_{Z_q}\left(\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q)\right), \tag{2.5.3}$$

where z_{ij} is the jth dimension of the ith point in $\boldsymbol{z}^{(1:q)}$. By the formula of g,

$$\begin{aligned}\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q) \quad = \quad & \frac{\partial}{\partial z_{ij}}\boldsymbol{\mu}^{(n)}(x^*(\text{before})) - \frac{\partial}{\partial z_{ij}}\boldsymbol{\mu}^{(n)}(x^*(\text{after})) \\ & -\frac{\partial}{\partial z_{ij}}\tilde{\sigma}_n(x^*(\text{after}), \boldsymbol{z}^{(1:q)})Z_q\end{aligned}$$

where $x^*(\text{before}) = \text{argmin}_{x\in\mathbb{A}}\boldsymbol{\mu}^{(n)}(x)$, $x^*(\text{after}) = \text{argmin}_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)})Z_q\right)$, and

$$\begin{aligned}\frac{\partial}{\partial z_{ij}}\tilde{\sigma}_n(x^*(\text{after}), \boldsymbol{z}^{(1:q)}) \quad = \quad & \left(\frac{\partial}{\partial z_{ij}}K^{(n)}(x^*(\text{after}), \boldsymbol{z}^{(1:q)})\right)(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\ & -K^{(n)}(x^*(\text{after}), \boldsymbol{z}^{(1:q)})(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\ & \left(\frac{\partial}{\partial z_{ij}}D^{(n)}(\boldsymbol{z}^{(1:q)})^T\right)(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1}.\end{aligned}$$

Now we can sample many times and take average to estimate the gradient of q-KG via (2.5.3). This technique is called infinitesimal perturbation analysis (IPA) in gradient estimation [40]. Since we can estimate the gradient of q-KG efficiently when $\mathbb{A}$ is finite, we will apply some standard gradient-based optimization algorithms, such as multi-start stochastic gradient ascent to maximize q-KG.

2.5.3 Approximating q-KG when $\mathbb{A}$ is infinite in (2.4.1) through discretization

We have specified how to maximize q-KG when $\mathbb{A}$ is finite in (2.4.1), but usually $\mathbb{A}$ is infinite. In this case, we will discretize $\mathbb{A}$ to approximate q-KG, and then

maximize over the approximate *q-KG*. The discretization itself is an interesting research topic [62].

In this chapter, the discrete set $\mathbb{A}_n$ is not chosen statically, but evolves over time: specifically, we suggest drawing M samples from the global optima of the posterior distribution of the Gaussian process (please refer to [23, 63] for a description of this technique). This sample set, denoted by $\mathbb{A}_n^M$, is then extended by the locations of previously sampled points $\boldsymbol{x}^{(1:n)}$ and the set of candidate points $\boldsymbol{z}^{(1:q)}$. Then (2.4.1) can be restated as

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}_n) = \min_{x \in \mathbb{A}_n} \mu^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \mathbb{A}_n} \mu^{(n+q)}(x) | \boldsymbol{z}^{(1:q)} \right], \tag{2.5.4}$$

where $\mathbb{A}_n = \mathbb{A}_n^M \cup \boldsymbol{x}^{(1:n)} \cup \boldsymbol{z}^{(1:q)}$. For the experimental evaluation we recompute $\mathbb{A}_n^M$ in every iteration after updating the posterior of the Gaussian process.

2.6 Numerical experiments

We conduct experiments in two different settings: the noise-free setting and the noisy setting. In both settings, we test the algorithms on well-known synthetic functions chosen from [3] and practical problems. Following previous literature [66], we use a constant mean prior and the ARD Matérn 5/2 kernel. In the noisy setting, we assume that $\sigma^2(x)$ is constant across the domain $\mathbb{A}$, and we estimate it together with other hyperparameters in the GP using maximum likelihood estimation (MLE). We set $M = 1000$ to discretize the domain following the strategy in Section 2.5.3. In general, the *q-KG* algorithm performs as well or better than state-of-art benchmark algorithms on both synthetic and real problems. It performs especially well in the noisy setting.

Before describing the details of the empirical results, we highlight the implementation details of our method and the open-source implementations of the benchmark methods. Our implementation inherits the open-source implementation of parallel EI from the `Metrics Optimization Engine` [75], which is fully implemented in `C++` with a python interface. We reuse their GP regression and GP hyperparameter fitting methods and implement the *q-KG* method in `C++`. Besides comparing to parallel EI in [75], we also compare our method to a well-known heuristic parallel EI implemented in `Spearmint` [28], the parallel UCB algorithm (GP-BUCB) and parallel UCB with pure exploration (GP-UCB-PE) both implemented in `Gpoptimization` [11].

2.6.1 Noise-free problems

In this section, we focus our attention on the noise-free setting, in which we can evaluate the objective exactly. We show that parallel knowledge gradient outperforms or is competitive with state-of-art benchmarks on several well-known test functions and tuning practical machine learning algorithms.

Synthetic functions

First, we test our algorithm along with the benchmarks on 4 well-known synthetic test functions: Branin2 on the domain $[-15, 15]^2$, Rosenbrock3 on the domain $[-2, 2]^3$, Ackley5 on the domain $[-2, 2]^5$, and Hartmann6 on the domain $[0, 1]^6$. We initiate our algorithms by randomly sampling $2d+2$ points from a Latin hypercube design, where d is the dimension of the problem. Figure 2.3 reports the mean and the standard deviation of the base 10 logarithm of the immediate regret by running

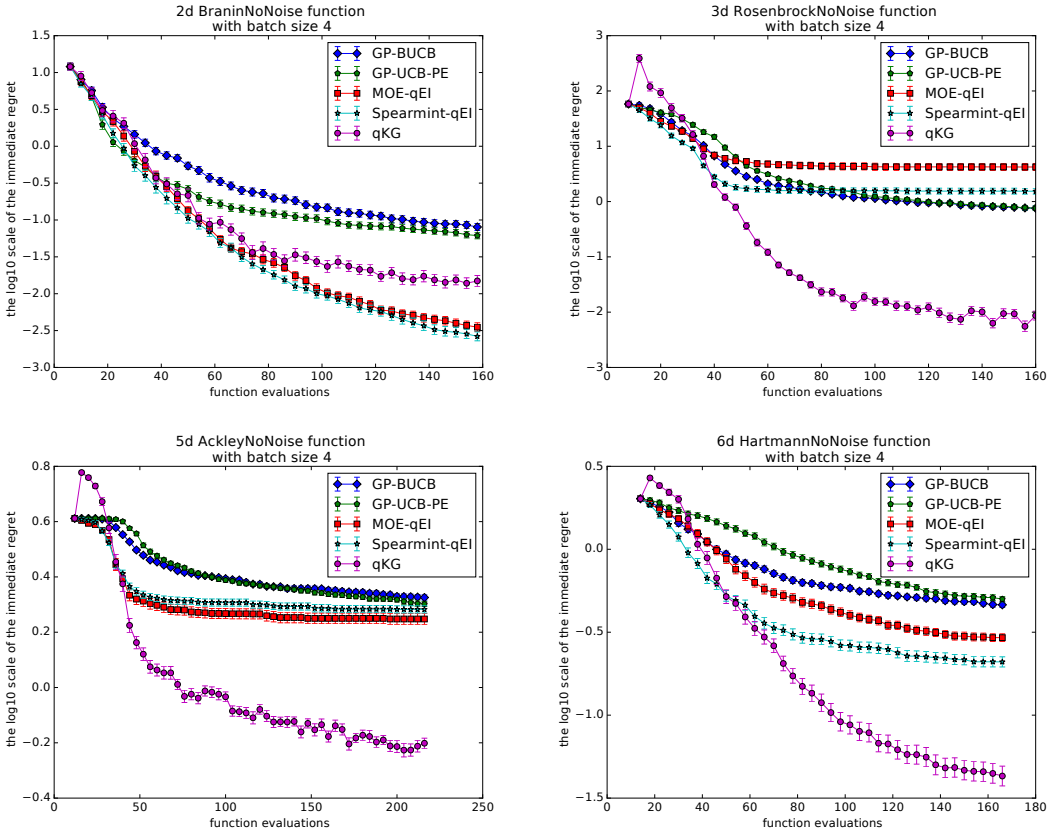

Figure 2.1: Performances on noise-free synthetic functions with $q = 4$. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of function evaluations.

100 random initializations with batch size $q = 4$.

The results show that *q-KG* is significantly better on Rosenbrock3, Ackley5 and Hartmann6, and is slightly worse than the best of the other benchmarks on Branin2. Especially on Rosenbrock3 and Ackley5, *q-KG* makes dramatic progress in early iterations.

Tuning logistic regression and convolutional neural networks (CNN)

In this section, we test the algorithms on two practical problems: tuning logistic regression on the MNIST dataset and tuning CNN on the CIFAR10 dataset. We

set the batch size to $q = 4$.

First, we tune logistic regression on the MNIST dataset. This task is to classify handwritten digits from images, and is a 10-class classification problem. We train logistic regression on a training set with 60000 instances with a given set of hyperparameters and test it on a test set with 10000 instances. We tune 4 hyperparameters: mini batch size from 10 to 2000, training iterations from 100 to 10000, the $\ell 2$ regularization parameter from 0 to 1, and learning rate from 0 to 1. We report the mean and standard deviation of the test error for 20 independent runs. From the results, one can see that both algorithms are making progress at the initial stage while *q-KG* can maintain this progress for longer and results in a better algorithm configuration in general.

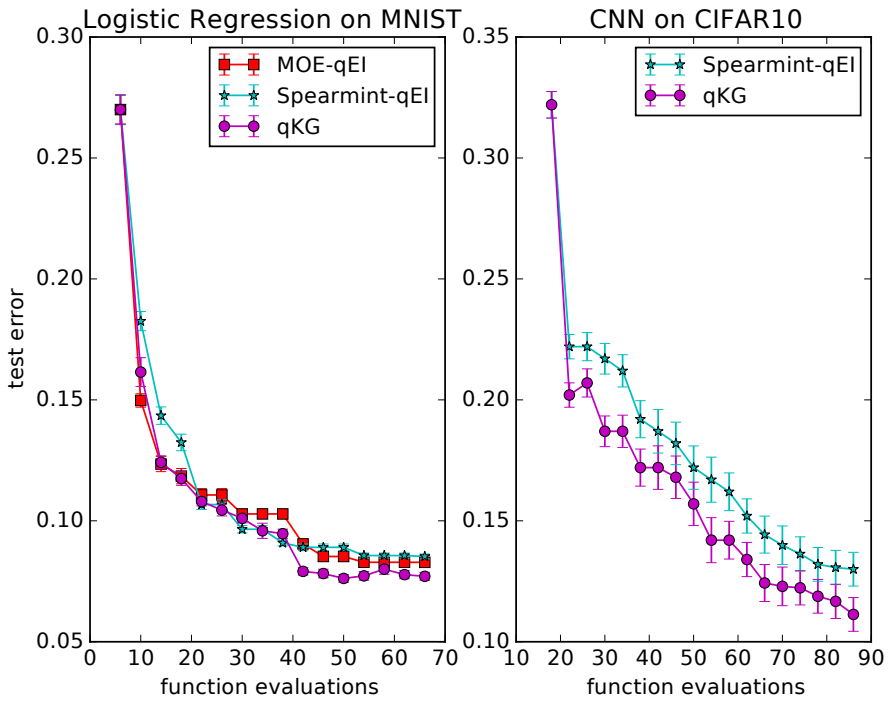

Figure 2.2: Performances on tuning machine learning algorithms with $q = 4$

In the second experiment, we tune a CNN on CIFAR10 dataset. This is also a 10-class classification problem. We train the CNN on the 50000 training data

with certain hyperparameters and test it on the test set with 10000 instances. For the network architecture, we choose the one in `tensorflow` tutorial. It consists of 2 convolutional layers, 2 fully connected layers, and on top of them is a softmax layer for final classification. We tune totally 8 hyperparameters: the mini batch size from 10 to 1000, training epoch from 1 to 10, the $\ell 2$ regularization parameter from 0 to 1, learning rate from 0 to 1, the kernel size from 2 to 10, the number of channels in convolutional layers from 10 to 1000, the number of hidden units in fully connected layers from 100 to 1000, and the dropout rate from 0 to 1. We report the mean and standard deviation of the test error for 5 independent runs. In this example, the *q-KG* is making better (more aggressive) progress than parallel EI even in the initial stage and maintain this advantage to the end. This architecture has been carefully tuned by the human expert, and achieve a test error around 14%, and our automatic algorithm improves it to around 11%.

2.6.2 Noisy problems

In this section, we study problems with noisy function evaluations. Our results show that the performance gains over benchmark algorithms from *q-KG* evident in the noise-free setting are even larger in the noisy setting.

Noisy synthetic functions

We test on the same 4 synthetic functions from the noise-free setting, and add independent gaussian noise with standard deviation $\sigma = 0.5$ to the function evaluation. The algorithms are not given this standard deviation, and must learn it from data.

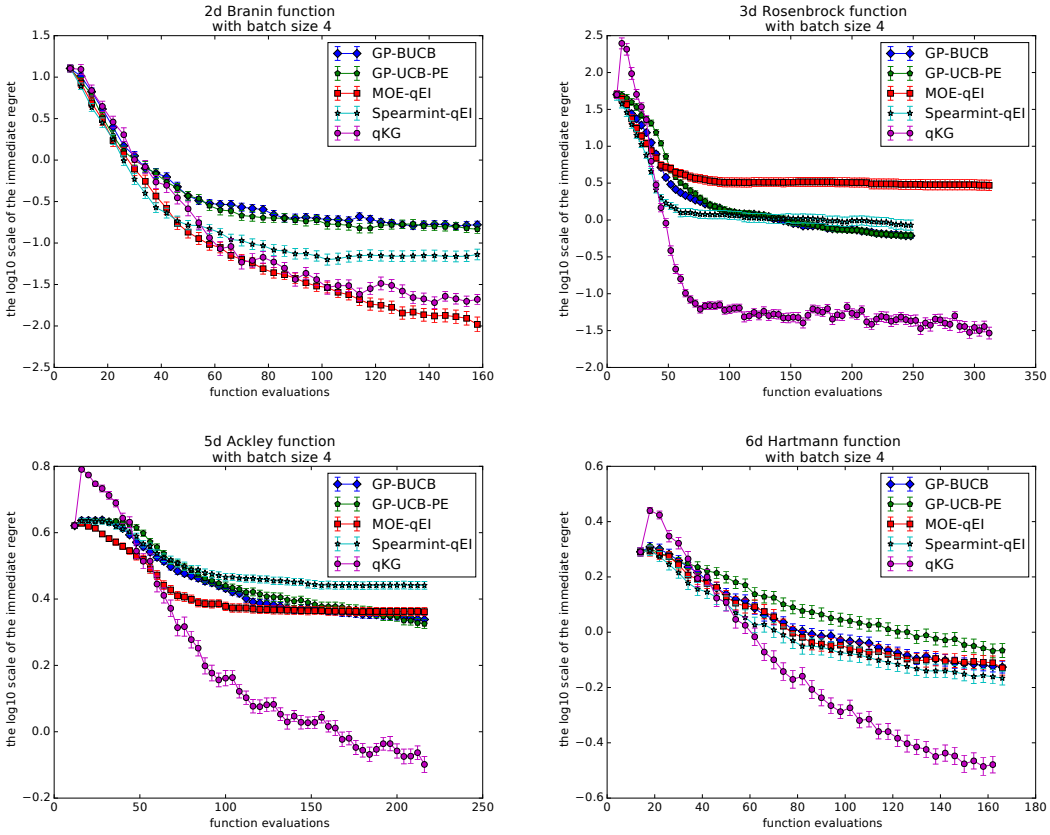

Figure 2.3: Performances on noisy synthetic functions with $q = 4$. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of function evaluations.

The results in Figure 2.4 show that *q-KG* is consistently better than or at least competitive with all competing methods. Also observe that the performance advantage of *q-KG* is larger than for noise-free problems.

Noisy logistic regression with small test sets

Testing on a large test set such as ImageNet is slow, especially when we must test many times for different hyperparameters. To speed up hyperparameter tuning, we may instead test the algorithm on a subset of the testing data to approximate the test error on the full set. We study the performance of our algorithm and

benchmarks in this scenario, focusing on tuning logistic regression on MNIST. We train logistic regression on the full training set of $60,000$, but we test the algorithm by testing on $1,000$ randomly selected samples from the test set, which provides a noisy approximation of the test error on the full test set.

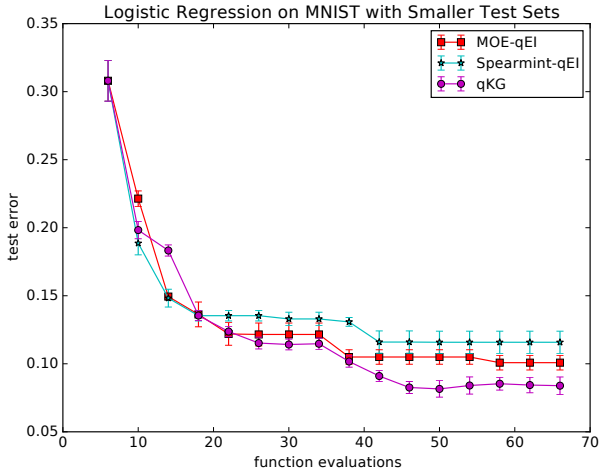

Figure 2.4: Tuning logistic regression on smaller test sets with $q = 4$

We report the mean and standard deviation of the test error on the full set using the hyperparameters recommended by each parallel BO algorithm for 20 independent runs. The result shows that *q-KG* is better than both versions of parallel EI, and its final test error is close to the noise-free test error (which is substantially more expensive to obtain). As we saw with synthetic test functions, *q-KG*'s performance advantage in the noisy setting is wider than in the noise-free setting.

2.7 Summary

In this chapter, we introduce a novel batch Bayesian optimization method *q-KG*, derived from a decision-theoretical perspective, and develop a computational method

to implement it efficiently. We show that *q-KG* outperforms or is competitive with the state-of-art benchmark algorithms on several synthetic functions and in tuning practical machine learning algorithms.

CHAPTER 3

DISCRETIZATION-FREE KNOWLEDGE GRADIENT METHODS FOR BAYESIAN OPTIMIZATION

3.1 Introduction

Knowledge gradient methods [13, 14] are widely studied for discrete R&S problems, which sample the one-step Bayes-optimal point. When used over continuous domains, previous work on the knowledge gradient [62, 84] often rely on a discretized finite approximation including the previous chapter. However, the discretization introduces error, and scales poorly as the dimension of domain grows. In this chapter, we develop a fast discretization-free parallel knowledge gradient method for Bayesian optimization. Our method is not restricted to the batch-sequential setting, but is useful in all settings where knowledge gradient can be used over continuous domains (such as fully sequential setting, multi-fidelity setting etc.).

3.2 Discretization-free computation of q-KG

Calculating and maximizing q-KG is difficult when $\mathbb{A}$ is continuous because the term $\min_{x\in\mathbb{A}} \mu^{(n+q)}(x)$ in Eq. (2.4.1) requires optimizing over an continuous domain, and then we must integrate this optimal value through its dependence on $y(\boldsymbol{z}^{(1:q)})$. Previous work on the knowledge gradient in continuous domains [60, 62, 84] overcomes this by taking minima within expectations not over the full domain $\mathbb{A}$ but over a discretized finite approximation. This supports analytic integration in [60, 62] and a sampling-based scheme in [84]. The discretization introduces error, and scales poorly as the dimension of $\mathbb{A}$ grows. In this section,

we provide the first efficient discretization-free strategy to maximize q-KG by a gradient-based optimizer. In Sect. 3.2.1, we describe how to compute q-KG by solving the inner optimization problem in Eq. (2.4.1). In Sect. 3.2.2, we provides an efficient unbiased estimator of the gradient of q-KG, which enables its fast optimization. Sect. 3.2.3 describes how we handle the hyperparameters of the GP model. Sect. 3.2.4 shows how to do asynchronous batch suggestion.

3.2.1 Estimating q-KG

Following [13], we express $\boldsymbol{\mu}^{(n+q)}(x)$ as

$$\begin{aligned}\boldsymbol{\mu}^{(n+q)}(x) &= \boldsymbol{\mu}^{(n)}(x) + K^{(n)}(x, \boldsymbol{z}^{(1:q)}) \left(K^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)})\right. \\ &\quad \left. +\mathrm{diag}\{\sigma^2(z_1), \cdots, \sigma^2(z_q)\}\right)^{-1} \left(y(\boldsymbol{z}^{(1:q)}) - \boldsymbol{\mu}^{(n)}(\boldsymbol{z}^{(1:q)})\right).\end{aligned}$$

Because $y(\boldsymbol{z}^{(1:q)}) - \boldsymbol{\mu}^{(n)}(\boldsymbol{z}^{(1:q)})$ is normally distributed with zero mean and covariance matrix $K^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) + \mathrm{diag}\{\sigma^2(z_1), \cdots, \sigma^2(z_q)\}$ with respect to the posterior after n observations, we can rewrite $\boldsymbol{\mu}^{(n+q)}(x)$ as

$$\boldsymbol{\mu}^{(n+q)}(x) = \boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) W_q, \tag{3.2.1}$$

where W_q is a standard q-dimensional normal random vector, and

$$\tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) = K^{(n)}(x, \boldsymbol{z}^{(1:q)})(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1},$$

where $D^{(n)}(\boldsymbol{z}^{(1:q)})$ is the Cholesky factor of the covariance matrix $K^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) + \mathrm{diag}\{\sigma^2(z_1), \cdots, \sigma^2(z_q)\}$. The gradient of $\boldsymbol{\mu}^{(n+q)}(x)$ with respect to x in (3.2.1) can also be computed by calculating the gradient of $\boldsymbol{\mu}^{(n)}(x)$ and $K^{(n)}(x, \boldsymbol{z}^{(1:q)})$, which means that $\min_{x \in \mathbb{A}} \mu^{(n+q)}(x)$ can be solved efficiently by an inner gradient-based optimizer. Now we can compute the q-KG factor using Monte Carlo sampling: we first sample W_q, optimize $\boldsymbol{\mu}^{(n+q)}(x)$ in (3.2.1) by an inner gradient-based optimizer, then plug in (2.4.1), repeat many times and take average.

3.2.2 Estimating the gradient of q-KG

Here we propose a novel method for calculating an unbiased estimator of the gradient of q-KG which we then use within stochastic gradient ascent to maximize q-KG. This method avoids discretization, and thus is "exact". It also improves speed significantly over a discretization-based scheme.

Exploiting (3.2.1), the q-KG factor can be expressed as

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \mathbb{E}_n \left[\min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q \right) \right].$$

Under sufficient regularity conditions [40], with details in Theorem 1 one can interchange the gradient and expectation operators,

$$\nabla q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = -\mathbb{E}_n \left[\nabla \min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q \right) \right]. \tag{3.2.2}$$

Since the multiplication, the inverse (when the inverse exists) and the Cholesky operators [65] preserve continuous differentiability, we have the following proposition.

Proposition 1. *When the mean function $\mu(\cdot)$ and the kernel function $K(\cdot)$ are continuous differentiable, $(x, \boldsymbol{z}^{(1:q)}) \mapsto \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q \right)$ is continuously differentiable.*

When $(x, \boldsymbol{z}^{(1:q)}) \mapsto \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q \right)$ is continuously differentiable and $\mathbb{A}$ is compact, the envelope theorem (Corollary 4 in [50]) implies

$$\nabla q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) = -\mathbb{E}_n \left[\nabla \left(\boldsymbol{\mu}^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q \right) \right],$$

where $x^*(W_q) \in \arg\min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q \right)$. This expression implies that $-\nabla \left(\boldsymbol{\mu}^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q \right)$ is an unbiased estimator

of $\nabla q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A})$. Then,

$$\nabla \left(\boldsymbol{\mu}^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q \right) \quad = \quad \nabla \tilde{\sigma}_n(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q$$

where

$$\begin{aligned} \nabla \tilde{\sigma}_n(x^*(W_q), \boldsymbol{z}^{(1:q)}) \quad = \quad & \left(\nabla K^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \right) (D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\ & -K^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) (D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\ & \left(\nabla D^{(n)}(\boldsymbol{z}^{(1:q)})^T \right) (D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1}. \end{aligned}$$

We can use this unbiased gradient estimator within stochastic gradient ascent [21], optionally with multiple starts, to solve the optimization problem (2.4.2). This technique is called infinitesimal perturbation analysis (IPA) in gradient estimation [40].

At the end of the section, we state the theorem which justifies the technique in the paper, whose proof is provided in Section 3.3.

Theorem 1. *When the mean function $\mu(\cdot)$ and the kernel function $K(\cdot)$ are continuously differentiable and the domain $\mathbb{A}$ is compact, the interchange of the gradient and expectation operators in (3.2.2) is valid.*

3.2.3 Bayesian Treatment of Hyperparameters.

We adopt a fully Bayesian treatment of hyperparameters similar to [66]. We draw m samples of hyperparameters $\phi^{(i)}$ for $1 \leq i \leq m$ via slice sampling [54] and average our acquisition function across them to obtain

$$q\text{-}KG_{\text{Integrated}}(\boldsymbol{z}^{(1:q)}) \quad = \quad \frac{1}{m} \sum_{i=1}^{m} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}; \phi^{(i)}), \tag{3.2.3}$$

where the additional argument $\phi^{(i)}$ in q-KG indicates that the computation is performed conditioning on hyperparameters $\phi^{(i)}$. In our experiments, we found this method to be computationally efficient and robust, although a more principled treatment of unknown hyperparameters within the knowledge gradient framework would instead marginalize over them when computing $\mu^{(n+q)}(x)$ and $\mu^{(n)}$.

3.2.4 Asynchronous q-KG Optimization

The (2.4.2) corresponds to the synchronous q-KG optimization, in which we wait for all q points from our previous batch to finish before searching for a new batch of q points. However, in some applications, we may wish to generate a new batch of points to evaluate next while $p(< q)$ points are still being evaluated, before we have their values. This is common in training machine learning algorithms, where different machine learning models do not necessarily finish at the same time.

We can generalize (2.4.2) to the asynchronous q-KG optimization. Given that p points are still under evaluation, now we would like to recommend a batch of $q - p$ points to evaluate. As we did for the synchronous q-KG optimization above, now we estimate the gradient of q-KG of the combined q points only with respect to the $q - p$ points that we need to recommend. Then we proceed the same way via gradient-based algorithms.

3.3 Proof of Theorem 1

In this section, we will prove Theorem 1. Recall that in Sect. 3.2, we have expressed the *q-KG* factor as follows,

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) \;=\; \mathbb{E}_n\left(g(\boldsymbol{z}^{(1:q)}, W_q)\right) \tag{3.3.1}$$

where the expectation is taken over W_q and

$$\begin{aligned} g(\boldsymbol{z}^{(1:q)}, W_q) &= \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)})W_q\right), \\ \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) &= K^{(n)}(x, \boldsymbol{z}^{(1:q)})(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1}. \end{aligned}$$

The main purpose of this section is to prove the following proposition, which is equivalent to Theorem 1 in Sect. 3.2.2.

Proposition 2. *When $\mu(\cdot)$ and $K(\cdot)$ are continuous differentiable,*

$$\frac{\partial}{\partial z_{ij}} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A})\Big|_{\boldsymbol{z}^{(1:q)}=\theta^{(1:q)}} \;=\; \mathbb{E}\left(\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, W_q)\right)\Big|_{\boldsymbol{z}^{(1:q)}=\theta^{(1:q)}}, \tag{3.3.2}$$

where $1 \le i \le q$, $1 \le j \le d$, z_{ij} is the j^{th} component of the i^{th} point in $\boldsymbol{z}^{(1:q)}$ and $\theta^{(1:q)} \in$ the interior of $\mathbb{A}^q$.

Without loss of generality, we assume that $\mathbb{A} = [0,1]^d$. Before proceeding, we define one more notation $f_{W_q}(z_{ij}) := g(\boldsymbol{z}^{(1:q)}, W_q)$ where $\boldsymbol{z}^{(1:q)}$ equals to $\theta^{(1:q)}$ component-wise except for z_{ij}. To prove Proposition **??**, we refer to Theorem 1 in [40], which requires three conditions to make (3.3.2) valid: there exists an open neighborhood $\Theta = (0,1)$ of θ_{ij} where θ_{ij} is the j^{th} component of the i^{th} point in $\theta^{(1:q)}$ such that (i) $f_{W_q}(z_{ij})$ is continuous everywhere in Θ for any fixed W_q, (ii) $f_{W_q}(z_{ij})$ is differentiable almost everywhere in Θ for any fixed W_q, (iii) the derivative of $f_{W_q}(z_{ij})$ (when it exists) is uniformly bounded by $\Gamma(W_q)$ for all $z_{ij} \in \Theta$, and the expectation of $\Gamma(W_q)$ is finite.

The proof of condition (i) and condition (ii). Under the condition that the mean function $\mu(\cdot)$ and the kernel function $K(\cdot)$ are continuous differentiable, by Proposition 1, $(x, z_{ij}) \mapsto \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)})W_q\right)$ is continuously differentiable. When $(x, z_{ij}) \mapsto \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)})W_q\right)$ is continuously differentiable and $\mathbb{A}$ is compact, the envelope theorem (Corollary 4 in [50]) implies (1) $f_{W_q}(z_{ij})$ is absolute continuous, i.e. differentiable almost everywhere; (2) the derivative of $f_{W_q}(z_{ij})$ (when it exists) is

$$\begin{aligned}\frac{\partial}{\partial z_{ij}} f_{W_q}(z_{ij}) &= -\frac{\partial}{\partial z_{ij}}\left(\boldsymbol{\mu}^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q\right) \\ &= -\frac{\partial}{\partial z_{ij}} \tilde{\sigma}^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q\end{aligned}$$

where $x^*(W_q) \in \arg\min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q\right)$. One can see that $f_{W_q}(z_{ij})$ is continuous everywhere and differentiable almost everywhere for any fixed W_q.

The proof of condition (iii). Given that

$$\frac{\partial}{\partial z_{ij}} f_{W_q}(z_{ij}) = -\frac{\partial}{\partial z_{ij}} \tilde{\sigma}^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)}) \cdot W_q$$

where $x^*(W_q) \in \arg\min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) \cdot W_q\right)$. Recall that from Sect. 2.5.2,

$$\begin{aligned}\frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(\boldsymbol{z}^{(1:q)}, x^*(W_q)) &= \left(\frac{\partial}{\partial z_{ij}} K^{(n)}(x^*(W_q)), \boldsymbol{z}^{(1:q)})\right)(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\ &\quad -K^{(n)}(x^*(W_q), \boldsymbol{z}^{(1:q)})(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\ &\quad \left(\frac{\partial}{\partial z_{ij}} D^{(n)}(\boldsymbol{z}^{(1:q)})^T\right)(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1}.\end{aligned}$$

By the result that $(z_{ij}, x) \to \frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(\boldsymbol{z}^{(1:q)}, x)$ is continuous, one can see that $\frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(\boldsymbol{z}^{(1:q)}, x^*(W_q))$ is bounded by a vector $0 \leq \Lambda < \infty$ as $\mathbb{A}$ is compact. Then $\left|\frac{d}{dz_{ij}} f_{W_q}(z_{ij})\right| \leq \sum_{i=1}^q \Lambda_i |w_i|$ where $W_q = (w_1, \cdots, w_q)^T$. And $\mathbb{E}\left(\sum_{i=1}^q \Lambda_i |w_i|\right) = \sqrt{2/\pi} \sum_{i=1}^q \Lambda_i < \infty$.

In addition, we can prove that the second moment of the gradient estimator is finite, which implies that SGA converges to a stationary point with an

appropriate sequence of step sizes. Since $|\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, W_q)| \leq \sum_{i=1}^{q} \Lambda_i |w_i|$, then $\mathbb{E}(\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, W_q))^2 \leq \sum_{i=1}^{q} \Lambda_i^2 + 4/\pi \sum_{i,j} \Lambda_i \Lambda_j < \infty$.

3.4 Summary

In this chapter, we propose a discretization-free method for optimizing the *q-KG* acquisition function proposed in Chapter 2. Our method is more accurate and faster than the previous discretization-based method, and can be used over all problem settings where KG is used over continuous domains.

CHAPTER 4

BAYESIAN OPTIMIZATION WITH GRADIENTS

4.1 Introduction

Bayesian optimization [4, 30] is able to find global optima with a remarkably small number of potentially noisy objective function evaluations. Bayesian optimization has thus been particularly successful for automatic hyperparameter tuning of machine learning algorithms [16, 17, 66, 72], where objectives can be extremely expensive to evaluate, noisy, and multimodal.

Bayesian optimization supposes that the objective function (e.g., the predictive performance with respect to some hyperparameters) is drawn from a prior distribution over functions, typically a Gaussian process (GP), maintaining a posterior as we observe the objective in new places. Acquisition functions, such as as expected improvement [24, 30, 58], upper confidence bound [71], predictive entropy search [23] or the knowledge gradient [62], determine a balance between exploration and exploitation, to decide where to query the objective next. By choosing points with the largest acquisition function values, one seeks to identify a global optimum using as few objective function evaluations as possible.

Bayesian optimization procedures do not generally leverage derivative information, beyond a few exceptions described in Sect. 4.2. By contrast, other types of continuous optimization methods [69] use gradient information extensively. The broader use of gradients for optimization suggests that gradients should also be quite useful in Bayesian optimization: (1) Gradients inform us about the objective's *relative* value as a function of location, which is well-aligned with optimization.

(2) In d-dimensional problems, gradients provide d distinct pieces of information about the objective's relative value in each direction, constituting $d+1$ values per query together with the objective value itself. This advantage is particularly significant for high-dimensional problems. (3) Derivative information is available in many applications at little additional cost. Recent work [e.g., 45] makes gradient information available for hyperparameter tuning. Moreover, in the optimization of engineering systems modeled by partial differential equations, which pre-dates most hyperparameter tuning applications [12], adjoint methods provide gradients cheaply [27, 59]. And even when derivative information is not readily available, we can compute approximative derivatives in parallel through finite differences.

In this paper, we explore the "what, when, and why" of Bayesian optimization with derivative information. We also develop a Bayesian optimization algorithm that effectively leverages gradients in hyperparameter tuning to outperform the state of the art. This algorithm accommodates incomplete and noisy gradient observations, can be used in both the sequential and batch settings, and can optionally reduce the computational overhead of inference by selecting the single most valuable directional derivatives to retain. For this purpose, we develop a new acquisition function, called the derivative-enabled knowledge-gradient (d-KG). This generalizes the previously proposed batch knowledge gradient method of [84] to the derivative setting, and replaces its approximate discretization-based method for calculating the knowledge-gradient acquisition function by a novel faster exact discretization-free method. We note that this discretization-free method is also of interest beyond the derivative setting, as it can be used to improve knowledge-gradient methods for other problem settings. We also provide a theoretical analysis of d-KG algorithm: it is one-step Bayes-optimal by construction when derivatives are available, and we show (1) that it provides one-step value greater than in the

derivative-free setting; and (2) that its estimator of the global optimum is asymptotically consistent. In numerical experiments we compare with state-of-the-art batch Bayesian optimization algorithms with and without derivative information, and the gradient-based optimizer BFGS with full gradients.

We assume familiarity with GPs and Bayesian optimization, for which we recommend [61] and [64] as a review. In Sect. 4.2 we begin by describing related work. In Sect. 4.3 we describe our Bayesian optimization algorithm exploiting derivative information. In Sect. 4.4 we compare the performance of our algorithm with several competing methods on a collection of synthetic and real problems.

4.2 Related Work

[56] proposes fully Bayesian optimization procedures that use derivative observations to improve the conditioning of the GP covariance matrix. Samples taken near previously observed points use only the derivative information to update the covariance matrix. Unlike our current work, derivative information is not fully utilized for optimization in this previous work in the sense that derivatives' availability do not affect the acquisition function. We directly compare with [56] within the KNN benchmark in Sect. 4.4.2.

[43, Sect. 4.2.1 and Sect. 5.2.4] incorporates derivatives into Bayesian optimization, modeling the derivatives of a GP as in [61, Sect. 9.4]. [43] shows that Bayesian optimization with the expected improvement (EI) acquisition function and complete gradient information at each sample can outperform BFGS. Our approach has six key differences: (i) we allow for noisy and incomplete derivative information; (ii) we develop a novel acquisition function that outperforms EI

with derivatives; (iii) we enable batch evaluations; (iv) we implement and compare batch Bayesian optimization with derivatives across several acquisition functions, on benchmarks and new applications such as kernel learning, logistic regression, deep learning and k-nearest neighbors, further revealing empirically where gradient information will be most valuable; (v) we provide a theoretical analysis of Bayesian optimization with derivatives; (vi) we develop a scalable implementation.

Very recently, [36] use GPs with derivative observations for minimum energy path calculations of atomic rearrangements and [1] studies expected improvement with gradient observations. In [1], a randomly selected directional derivative is retained in each iteration for computational reasons, which is similar to our approach of retaining a single directional derivative, though differs in its random selection in contrast with our value-of-information-based selection. Our approach is complementary to these works.

For batch Bayesian optimization, several recent algorithms have been proposed that choose a set of points to evaluate in each iteration [6, 10, 19, 34, 49, 63, 66, 76]. Within this area, our approach to handling batch observations is most closely related to the batch knowledge gradient (q-KG) of [84]. We generalize this approach to the derivative setting, and provide a novel exact method for computing the knowledge-gradient acquisition function that avoids the discretization used in [84]. This generalization improves speed and accuracy, and is also applicable to other knowledge gradient methods in continuous search spaces.

Recent advances improving both access to derivatives and computational tractability of GPs make Bayesian optimization with gradients increasingly practical and timely for discussion.

4.3 Knowledge Gradient with Derivatives

Sect. 4.3.1 reviews a general approach to incorporating derivative information into GPs for Bayesian optimization. Sect. 4.3.2 introduces a novel acquisition function *d-KG*, based on the knowledge gradient approach, which utilizes derivative information. Sect. 4.3.3 computes this acquisition function and its gradient efficiently using a novel fast discretization-free approach. Sect. 4.3.4 shows that this algorithm provides greater value of information than in the derivative-free setting, is one-step Bayes-optimal, and is asymptotically consistent when used over a discretized feasible space.

4.3.1 Derivative Information

Given an expensive-to-evaluate function f, we wish to find $\operatorname{argmin}_{x\in\mathbb{A}} f(x)$, where $\mathbb{A} \subset \mathbb{R}^d$ is the domain of optimization. We place a GP prior over $f\colon \mathbb{A} \to \mathbb{R}$, which is specified by its mean function $\mu\colon \mathbb{A} \to \mathbb{R}$ and kernel function $K\colon \mathbb{A} \times \mathbb{A} \to \mathbb{R}$. We first suppose that for each sample of x we observe the function value and all d partial derivatives, possibly with independent normally distributed noise, and then later show discuss relaxation to observing only a single directional derivative.

Since the gradient is a linear operator, the gradient of a GP is also a GP (see also Sect. 9.4 in [61]), and the function and its gradient follows a multi-output GP with mean function $\tilde{\mu}$ and kernel function $\tilde{K}$ defined below:

$$\tilde{\mu}(x) = (\mu(x), \nabla\mu(x))^T, \quad \tilde{K}(x, x') = \begin{pmatrix} K(x, x') & J(x, x') \\ J(x', x)^T & H(x, x') \end{pmatrix} \tag{4.3.1}$$

where $J(x, x') = \left(\frac{\partial K(x,x')}{\partial x'_1}, \cdots, \frac{\partial K(x,x')}{\partial x'_d}\right)$ and $H(x, x')$ is the $d \times d$ Hessian of $K(x, x')$.

When evaluating at a point x, we observe the noise-obscured function value $y(x)$ and gradient $\nabla y(x)$. Jointly, these observations form a $(d+1)$-dimensional vector with conditional distribution

$$(y(x), \nabla y(x))^T \mid f(x), \nabla f(x) \sim \mathcal{N}\left(\left(f(x), \nabla f(x)\right)^T, \mathrm{diag}(\sigma^2(x))\right), \quad (4.3.2)$$

where $\sigma^2 : \mathbb{A} \to \mathbb{R}^{d+1}_{\geq 0}$ gives the variance of the observational noise. If σ^2 is not known, we may estimate it from data. The posterior distribution is again a GP. We refer to the mean function of this posterior GP after n samples as $\tilde{\mu}^{(n)}(\cdot)$ and its kernel function as $\tilde{K}^{(n)}(\cdot, \cdot)$. Their formulae are given in the supplementary material for completeness.

If some entries of $(y(x), \nabla y(x))$ are not observed, then the conditional distribution of the remaining vector is obtained by omitting the corresponding entries from $(f(x), \nabla f(x))$ and $\sigma^2(x)$ in (4.3.2). Moreover, the vector remaining after these entries are removed from $(f(x), \nabla f(x))$ still obeys a multivariate normal distribution, with parameters obtained by omitting the corresponding entries of the mean vector and covariance matrix from (4.3.1). This allows performing inference similarly to the procedure described above for full gradient observations.

4.3.2 The *d-KG* Acquisition Function

We propose a novel Bayesian optimization algorithm to exploit available derivative information, based on the knowledge gradient approach [13]. We call this algorithm the *derivative-enabled knowledge gradient* (*d-KG*).

The algorithm proceeds iteratively, selecting in each iteration a batch of q points in $\mathbb{A}$ that has a maximum *value of information* (VOI). Suppose we have observed n points, and recall from Sect. 4.3.1 that $\tilde{\mu}^{(n)}(x)$ is the $(d+1)$-dimensional vector giving the posterior mean for $f(x)$ and its d partial derivatives at x. Sect. 4.3.1 discusses how to remove the assumption that all $d+1$ values are provided.

The expected value of $f(x)$ under the posterior distribution is $\tilde{\mu}_1^{(n)}(x)$. If after n samples we were to make an irrevocable (risk-neutral) decision now about the solution to our overarching optimization problem and receive a loss equal to the value of f at the chosen point, we would choose $\operatorname{argmin}_{x\in\mathbb{A}}\tilde{\mu}_1^{(n)}(x)$ and suffer conditional expected loss $\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(n)}(x)$. Similarly, if we made this decision after $n+q$ samples our conditional expected loss would be $\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(n+q)}(x)$. Therefore, we define the *d-KG* factor for a given set of q candidate points $\boldsymbol{z}^{(1:q)}$ as

$$d\text{-}KG(\boldsymbol{z}^{(1:q)},\mathbb{A}) \quad = \quad \min_{x\in\mathbb{A}}\tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n\left[\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(n+q)}(x) \;\middle|\; \boldsymbol{z}^{(1:q)}\right], \tag{4.3.3}$$

where $\mathbb{E}_n[\cdot]$ is the expectation taken with respect to the posterior distribution after n evaluations, and the distribution of $\tilde{\mu}_1^{(n+q)}(\cdot)$ under this posterior marginalizes over the observations $\left(y(\boldsymbol{z}^{(1:q)}), \nabla y(\boldsymbol{z}^{(1:q)})\right) = \left(y(z^{(i)}), \nabla y(z^{(i)}) : i = 1,\ldots,q\right)$ upon which it depends. We subsequently refer to Eq. (4.3.3) as the *inner optimization problem.*

The *d-KG* algorithm then seeks to evaluate the batch of points next that maximizes the *d-KG* factor,

$$\max_{\boldsymbol{z}^{(1:q)}\subset\mathbb{A}} d\text{-}KG(\boldsymbol{z}^{(1:q)},\mathbb{A}). \tag{4.3.4}$$

We refer to Eq. (4.3.4) as the *outer optimization problem.* *d-KG* solves the outer approximation problem using the method described in Sect. 4.3.3.

The *d-KG* acquisition function differs from the batch knowledge gradient acquisition function in [84] because here the time-$n+q$ posterior mean $\tilde{\mu}_1^{(n+q)}(x)$ depends on $\nabla y(\boldsymbol{z}^{(1:q)})$. This in turn requires calculating the pre-posterior distribution of these gradient observations under the time-n posterior and marginalizing over them. Thus, the *d-KG* algorithm differs from *q-KG* not just in that gradient observations change the posterior, but also in that the prospect of *future* gradient observations changes the acquisition function. It also differs from [84] by using a novel discretization-free method for computing the acquisition function (see Sect. 4.3.3).

Fig. 4.1 illustrates the behavior of *d-KG* and *d-EI* on a 1-d example. The latter generalizes expected improvement (EI) to batch-acquisition with derivative information [43]. *d-KG* clearly chooses a better point to evaluate than *d-EI*.

Including all d partial derivatives can be computationally prohibitive since GP inference scales as $\mathcal{O}(n^3(d+1)^3)$. To overcome this challenge while retaining the value of derivative observations, we can include only one directional derivative from each iteration in our inference. *d-KG* can naturally decide which derivative to include, and can adjust our choice of where to best sample given that we observe more limited information. We define the *d-KG* acquisition function for observing only the function value and the derivative with direction θ at $\boldsymbol{z}^{(1:q)}$ as

$$d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A}) = \min_{x \in \mathbb{A}} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n \left[\min_{x \in \mathbb{A}} \tilde{\mu}_1^{(n+q)}(x) \;\middle|\; \boldsymbol{z}^{(1:q)}, \theta \right]. \tag{4.3.5}$$

where conditioning on θ is here understood to mean that $\tilde{\mu}_1^{(n+q)}(x)$ is the conditional mean of $f(x)$ given $y(\boldsymbol{z}^{(1:q)})$ and $\theta \cdot \nabla y(\boldsymbol{z}^{(1:q)}) = (\theta \cdot \nabla y(\boldsymbol{z}^{(i)}) : i = 1, \ldots, q)$. The full algorithm is as follows.

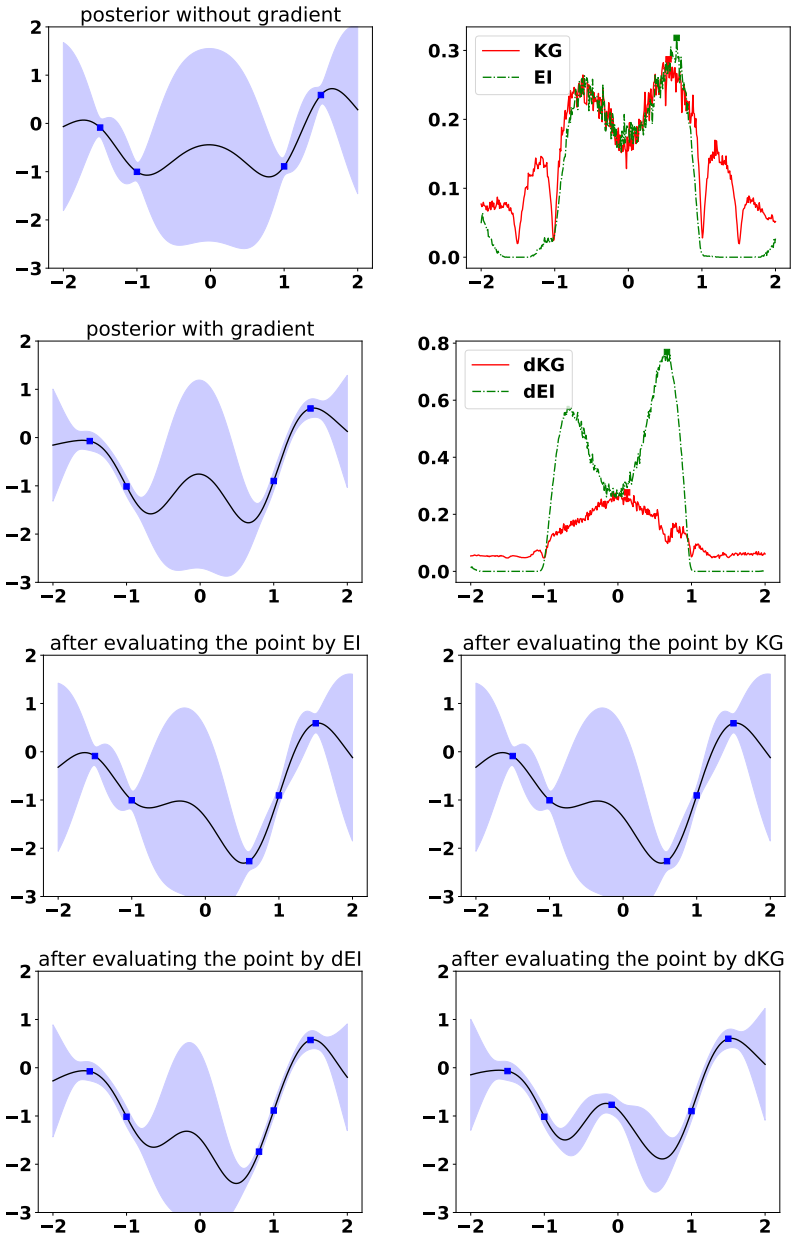

Figure 4.1: The topmost plots show (1) the posterior surfaces of a function sampled from a one dimensional GP with and without incorporating observations of the gradients. Note that the posterior variance is smaller if the gradients are incorporated; (2) the utility of sampling each point under the value of information criteria of KG and EI in both settings. If no derivatives are observed, both KG and EI will query a point with high potential gain (i.e. a small expected function value). On the other hand, when gradients are observed, *d-KG* makes a considerably better sampling decision, whereas *d-EI* samples essentially the same location as *EI*. The plots in the bottom row depict the posterior surface *after* the respective sample. Interestingly, KG benefits more from observing the gradients than EI (the last two plots): *d-KG* samples a point whose observation yields an accurate knowledge of the location of the optimum, while *d-EI* still has considerable uncertainty around the optimum.

Algorithm 2 *d-KG* with Relevant Directional Derivative Detection

1: **for** $t = 1$ to N **do**
2: $(\boldsymbol{z}^{(1:q)^*}, \theta^*) = \text{argmax}_{\boldsymbol{z}^{(1:q)},\theta} d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A})$
3: Augment data with $y(\boldsymbol{z}^{(1:q)^*})$ and $\theta^{*T}\nabla y(\boldsymbol{z}^{(1:q)^*})$. Update our posterior on $(f(x), \nabla f(x))$.
4: **end for**
Return $x^* = \text{argmin}_{x\in\mathbb{A}} \tilde{\mu}_1^{Nq}(x)$

4.3.3 Efficient Exact Computation of *d-KG*

Calculating and maximizing *d-KG* is difficult when $\mathbb{A}$ is continuous because the term $\min_{x\in\mathbb{A}} \tilde{\mu}_1^{(n+q)}(x)$ in Eq. (4.3.5) requires optimizing over an continuous domain, and then we must integrate this optimal value through its dependence on $y(\boldsymbol{z}^{(1:q)}), \theta \cdot \nabla y(\boldsymbol{z}^{(1:q)})$. Previous work on the knowledge gradient in continuous domains [60, 62, 84] overcomes this by taking minima within expectations not over the full domain $\mathbb{A}$ but over a discretized finite approximation. This supports analytic integration in [60, 62] and a sampling-based scheme in [84]. The discretization introduces error, and scales poorly as the dimension of $\mathbb{A}$ grows.

Here we propose a novel method for calculating an unbiased estimator of the gradient of *d-KG* which we then use within stochastic gradient ascent to maximize *d-KG*. This method avoids discretization, and thus is "exact". It also improves speed significantly over a discretization-based scheme.

With details in the supplement (Sect. B.2), the *d-KG* factor can be expressed as

$$d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A}) = \mathbb{E}_n\left[\min_{x\in\mathbb{A}} \hat{\mu}_1^{(n)}(x) - \min_{x\in\mathbb{A}}\left(\hat{\mu}_1^{(n)}(x) + \hat{\sigma}^{(n)}(x, \theta, \boldsymbol{z}^{(1:q)}) \cdot W\right)\right],$$

where $\hat{\mu}$ is the mean function of $(f(x), \theta^T\nabla f(x))$ after n evaluations, W is a $2q$ dimensional standard normal random vector and $\hat{\sigma}^{(n)}(x, \theta, \boldsymbol{z}^{(1:q)})$ is related to the

kernel function of $(f(x), \theta^T \nabla f(x))$ after n evaluations with an exact form specified in (B.2.2) of the supplement.

Under sufficient regularity conditions [41] one can interchange the gradient and expectation operators,

$$\nabla d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A}) = -\mathbb{E}_n \left[\nabla \min_{x \in \mathbb{A}} \left(\hat{\mu}_1^{(n)}(x) + \hat{\sigma}^{(n)}(x, \theta, \boldsymbol{z}^{(1:q)}) \cdot W \right) \right].$$

When $(x, \boldsymbol{z}^{(1:q)}) \mapsto \left(\hat{\mu}_1^{(n)}(x) + \hat{\sigma}^{(n)}(x, \theta, \boldsymbol{z}^{(1:q)}) \cdot W \right)$ is continuously differentiable and $\mathbb{A}$ is compact, the envelope theorem [50] implies

$$\nabla d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A}) = -\mathbb{E}_n \left[\nabla \left(\hat{\mu}_1^{(n)}(x^*(W)) + \hat{\sigma}^{(n)}(x^*(W), \theta, \boldsymbol{z}^{(1:q)}) \cdot W \right) \right],$$

where $x^*(W) \in \arg\min_{x \in \mathbb{A}} \left(\hat{\mu}_1^{(n)}(x) + \hat{\sigma}^{(n)}(x, \theta, \boldsymbol{z}^{(1:q)}) \cdot W \right)$.

This expression implies that $\nabla \left(\hat{\mu}_1^{(n)}(x^*(W)) + \hat{\sigma}^{(n)}(x^*(W), \theta, \boldsymbol{z}^{(1:q)}) \cdot W \right)$ is an unbiased estimator of $\nabla d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A})$. We can use this unbiased gradient estimator within stochastic gradient ascent [21], optionally with multiple starts, to solve the outer optimization problem (4.3.4).

Bayesian Treatment of Hyperparameters. We adopt a fully Bayesian treatment of hyperparameters similar to [66]. We draw m samples of hyperparameters $\phi^{(i)}$ for $1 \leq i \leq m$ via slice sampling [54] and average our acquisition function across them to obtain

$$d\text{-}KG_{\text{Integrated}}(\boldsymbol{z}^{(1:q)}, \theta) \quad = \quad \frac{1}{m} \sum_{i=1}^{m} d\text{-}KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A}; \phi^{(i)}), \tag{4.3.6}$$

where the additional argument $\phi^{(i)}$ in d-KG indicates that the computation is performed conditioning on hyperparameters $\phi^{(i)}$. In our experiments, we found this method to be computationally efficient and robust, although a more principled treatment of unknown hyperparameters within the knowledge gradient framework would instead marginalize over them when computing $\tilde{\mu}^{(n+q)}(x)$ and $\tilde{\mu}^{(n)}$.

4.3.4 Theoretical Analysis

Here we present three theoretical results giving insight into the properties of *d-KG*. (The proofs have been deferred to the supplement due to space constraints.) For the sake of simplicity, we suppose all partial derivatives are provided to *d-KG*. Note that similar results hold for *d-KG* with relevant directional derivative detection. We begin by stating that the value of information (VOI) obtained by *d-KG* exceeds the VOI that can be achieved in the derivative-free setting.

Proposition 3. *Given identical posteriors* $\tilde{\mu}^{(n)}$*,*

$$d\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) \geq q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}),$$

where q-KG is the batch knowledge gradient acquisition function without gradients proposed by [84].

Next, we show that *d-KG* is one-step Bayes-optimal by construction.

Proposition 4. *If only one iteration is left and we can observe both function values and its partial derivatives, then d-KG is Bayes-optimal among all feasible policies.*

As a complement to one-step optimality, we show that *d-KG* is asymptotically consistent when the feasible set $\mathbb{A}$ is finite. Asymptotic consistency means that *d-KG* will choose the correct solution when the number of iterations goes to infinity.

Theorem 2. *The d-KG algorithm is asymptotically consistent, i.e.*

$$\lim_{N \to \infty} f(x^*(d\text{-}KG, N)) = \min_{x \in \mathbb{A}} f(x)$$

almost surely, where $x^*(d\text{-}KG, N)$ *is the point recommended by d-KG after N iterations.*

4.4 Experiments

We evaluate the performance of the proposed algorithm *d-KG* with relevant directional derivative detection (Algorithm 1) on six standard synthetic benchmarks (see Fig. 4.2 and Sect. 4 in the supplement). Moreover, we examine its ability to tune the hyperparameters for the weighted k-nearest neighbor metric, logistic regression, deep learning, and for a spectral mixture kernel (see Fig. 4.3). We will provide an easy-to-use `Python` package with the core written in `C++`. We compare *d-KG* to several state-of-the-art methods: (1) The batch expected improvement method (*EI*) of [76] that does not utilize derivative information and an extension of *EI* that incorporates derivative information denoted by *d-EI*. Note that *d-EI* is similar to [43] but handles incomplete gradients and supports batches. (2) The batch GP-UCB-PE method of [6] that does not utilize derivative information, and an extension of this method that does. (3) The batch knowledge gradient algorithm without derivative information (*q-KG*) of [84]. Moreover, we generalize the method of [56] to batches and evaluate it on the KNN benchmark. We stress that all of the above algorithms can be run even if not all partial derivatives are given. In benchmarks that provide the full gradient, we additionally compare to the gradient-based method L-BFGS-B provided in `scipy`. We suppose that the objective function f is drawn from a Gaussian process $GP(\mu, \Sigma)$, where μ is a constant mean function and Σ is the squared exponential kernel. We sample $K = 20$ sets of hyperparameters by slice sampling.

Recall that the immediate regret is defined as the loss with respect to a global optimum. The plots for synthetic benchmark functions, shown in Fig. 4.2, report the log10 scale of immediate regret of the solution that each algorithm would pick as a function of the number of function evaluations. For the other experiments

the plots depict the objective value of the solution instead of the immediate regret. The error bars give the mean value plus and minus one standard deviation. The number of replications is stated in each benchmark's description.

4.4.1 Synthetic Test Functions

We evaluate all methods on four test functions chosen from [3]. To examine the gain from *noisy derivative information*, we sample additive normally distributed noise with mean zero and standard deviation $\sigma = 0.5$ for both the objective function and its partial derivatives. Note that σ is not provided to the algorithms and thus estimated from observations. Moreover, we investigate how the performance of the algorithms is affected if partial derivatives are not given for all parameters. We also experiment with two different batch sizes: we use a batch size $q = 4$ for the Branin, and Rosenbrock functions; otherwise, we use a batch size $q = 8$. For 2d Branin on domain $[-15, 15]^2$ and 6d Hartmann function on $[0, 1]^6$, we assume that the full gradient is available. The 4d Levy benchmark is on $[-10, 10]^4$, where the fourth partial derivative is observable with noise. For the 8d Cosine mixture function on $[-1, 1]^8$ we provide noisy 1st and 2nd partial derivatives. The experimental results are summarized in Fig. 4.2. Summing up, we see that *d-KG* successfully exploits noisy derivative information and has the best overall performance.

4.4.2 Real-World Test Functions

Weighted k-Nearest Neighbor. Suppose a cab company wishes to predict the duration of trips. Clearly, the duration not only depends on the endpoints of the trip, but also on the day and time. In this benchmark we tune a weighted

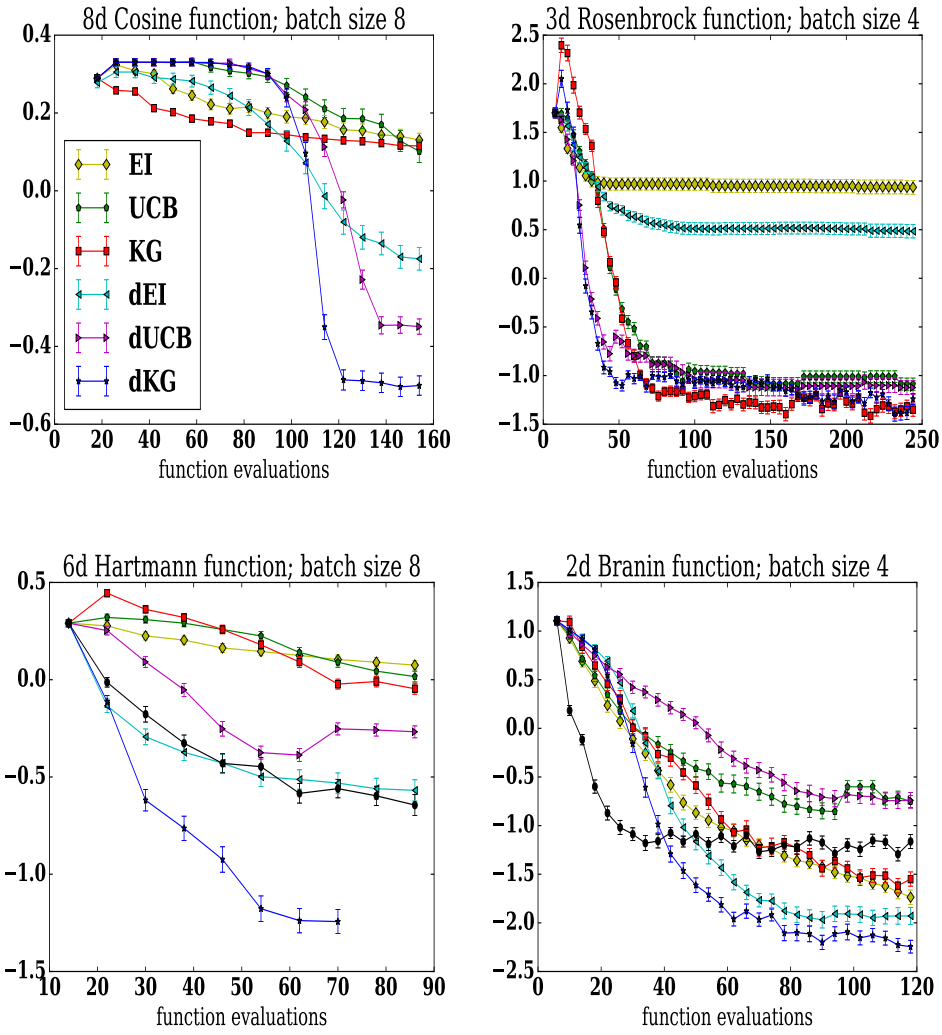

Figure 4.2: The average performance of 100 replications (the log10 of the immediate regret vs. the number of function evaluations). *d-KG* performs significantly better than its competitors for all benchmarks. In Branin and Hartmann, we also plot black lines, which is the performance of BFGS.

k-nearest neighbor (KNN) metric to optimize predictions of these durations, based on historical data. A trip is described by the pick-up time t, the pick-up location (p_1, p_2), and the drop-off point (d_1, d_2). Then the estimate of the duration is obtained as a weighted average over all trips $D_{m,t}$ in our database that happened in the time interval $t \pm m$ minutes, where m is a tunable hyperparameter: $\text{Prediction}(t, p_1, p_2, d_1, d_2) = (\sum_{i \in D_{m,t}} \text{duration}_i \times \text{weight}(i))/(\sum_{i \in D_{m,t}} \text{weight}(i))$. The weight of trip $i \in D_{m,t}$ in this prediction is given by $\text{weight}(i) =$

$$((t-t^i)^2/l_1^2 + (p_1-p_1^i)^2/l_2^2 + (p_2-p_2^i)^2/l_3^2 + (d_1-d_1^i)^2/l_4^2 + (d_2-d_2^i)^2/l_5^2)^{-1},$$

where $(t^i, p_1^i, p_2^i, d_1^i, d_2^i)$ are the respective parameter values for trip i, and $(l_1, l_2, l_3, l_4, l_5)$ are tunable hyperparameters. Thus, we have 6 hyperparameters to tune: $(m, l_1, l_2, l_3, l_4, l_5)$. We choose m in $[30, 200]$, l_1^2 in $[10^1, 10^8]$, and $l_2^2, l_3^2, l_4^2, l_5^2$ each in $[10^{-8}, 10^{-1}]$.

We use the yellow cab NYC public data set from June **(year?)**, sampling 10000 records from June 1 – 25 as training data and 1000 trip records from June 26 – 30 as validation data. Our test criterion is the root mean squared error (RMSE), for which we compute the partial derivatives on the validation dataset with respect to the hyperparameters $(l_1, l_2, l_3, l_4, l_5)$, while the hyperparameter m is not differentiable. The experimental results in Fig. 4.3 demonstrate that *d-KG* outperforms the other algorithms eventually. For UCB and KG acquisition functions, exploiting derivative information provides an advantage.

Kernel Learning. Spectral mixture kernels [80] can be used for flexible kernel learning to enable long-term extrapolation. These kernels are obtained by modeling a spectral density by a mixture of Gaussians. While any stationary kernel can be described by a spectral mixture kernel with a particular setting of its hyperparameters, initializing and learning these parameters can be difficult. Although we have access to an analytic closed form of the (marginal likelihood) objective, this function is (i) expensive to evaluate and (ii) highly multimodal. Moreover, (iii) derivative information is available. Thus, learning flexible kernel functions is a perfect candidate for our approach.

The task is to train a 3-component spectral mixture kernel on an airline data set [80]. We have to determine the mixture weights, means, and variances, for each

of the three Gaussians. Fig. 4.3 summarizes the experimental performances for batch size $q = 8$. BFGS turns out to be sensitive to its initialization and human intervention and is often trapped in local optima. *d-KG*, on other hand, more consistently finds a good solution, and obtains the best solution of all algorithms (within the step limit). Overall, we observe that gradient information is highly valuable in performing this kernel learning task.

Logistic Regression and Deep Learning. We tune logistic regression and a feedforward neural network with 2 hidden layers on the MNIST dataset [39], a standard classification task for handwritten digits. The training set contains 60000 images, the test set 10000. We tune 4 hyperparameters for logistic regression: the $\ell 2$ regularization parameter from 0 to 1, learning rate from 0 to 1, mini batch size from 20 to 2000 and training epochs from 5 to 50. The first derivatives of the first two parameters are can be obtained via the technique of [45]. For the neural network, we additionally tune the number of hidden units in $[50, 500]$.

Fig. 4.3 reports the mean and standard deviation of the mean cross-entropy loss on the test set for 20 replications. *d-KG* outperforms the other approaches, which suggests that derivative information is helpful. Our algorithm proves its value in tuning a deep neural network, which harmonizes with research computing the gradient of hyperparameters [45, 57].

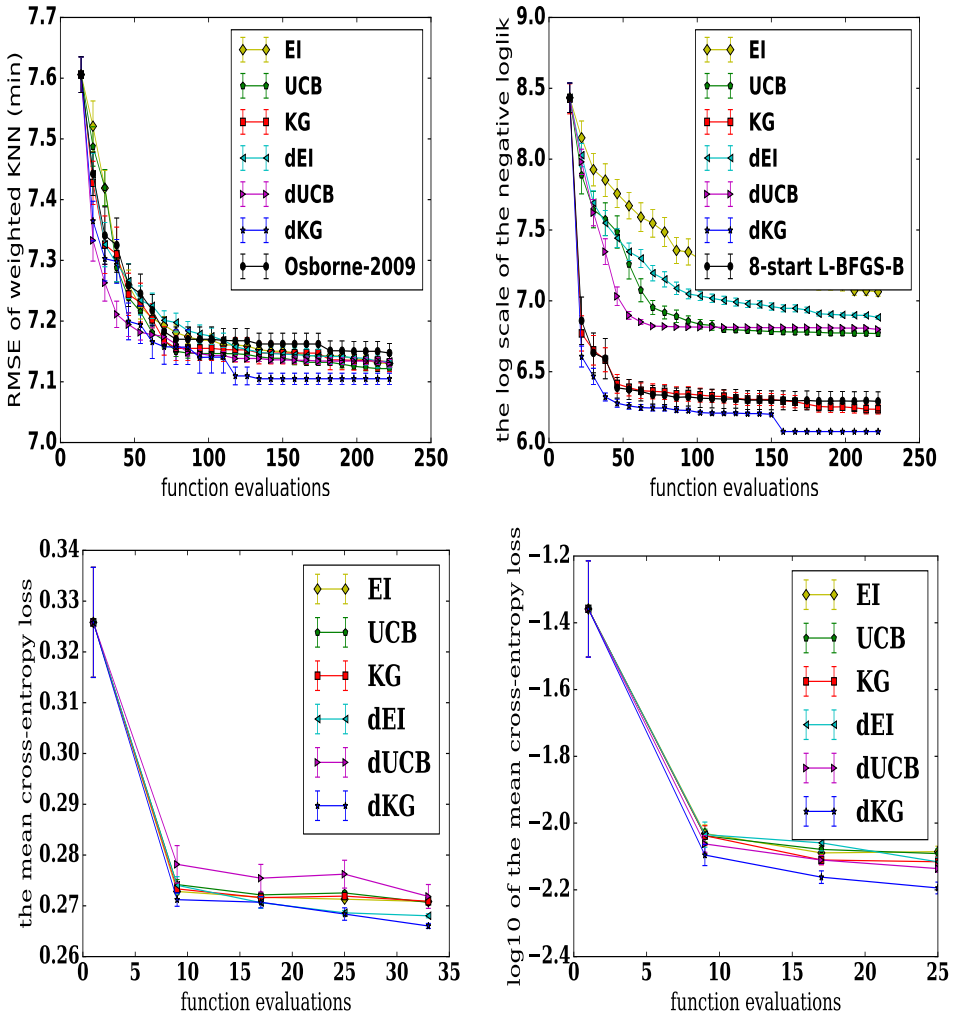

Figure 4.3: Results for the weighted KNN benchmark, the spectral mixture kernel benchmark, logistic regression and deep neural network (from left to right), all with batch size 8 and averaged over 20 replications.

4.5 Summary

Bayesian optimization is primarily applied to low dimensional problems where we wish to find a good solution with a very small number of objective function evaluations. We considered several such benchmarks, as well as logistic regression, deep learning, kernel learning, and k-nearest neighbor applications. We have shown that

in this context derivative information can be extremely useful: we can greatly decrease the number of objective function evaluations, especially when building upon the knowledge gradient acquisition function, even when derivative information is noisy and only available for some variables.

Bayesian optimization is increasingly being used to automate parameter tuning in machine learning, where objective functions can be extremely expensive to evaluate. For example, the parameters could even represent the hyperparameters of a deep neural network. We expect derivative information with Bayesian optimization to help enable such promising applications, moving us towards fully automatic and principled approaches to statistical machine learning.

In the future, one could combine derivative information with flexible deep projections [82], and recent advances in scalable Gaussian processes for $\mathcal{O}(n)$ training and $\mathcal{O}(1)$ test time predictions [81, 83]. These steps would help make Bayesian optimization applicable to a much wider range of problems, wherever standard gradient based optimizers are used – even when we have analytic objective functions that are *not* expensive to evaluate – while retaining faster convergence and robustness to multimodality.

CHAPTER 5

CONTINUOUS-FIDELITY KNOWLEDGE GRADIENT FOR FAST BAYESIAN OPTIMIZATION

5.1 Introduction

In lots of applications, there exists some low-fidelity but cheap approximations, e.g.

- When tuning hyperparameters of a machine learning model, we can evaluate the configuration by training on smaller training data, and/or with less training iterations.
- When conducting optimization via a Monte Carlo simulator, we can evaluate the simulation configuration by running with less replications.
- When optimizing an engineering system modeled by a PDE, we can solve the PDE with a less fine grid.

Recently, some principled ways of exploiting these low-fidelity but cheap approximations are proposed both in and out of the field of Bayesian optimization. [42] shows that a principled early-stopping method called `HYPERBAND` can outperform conventional Bayesian optimization when tuning hyperparameters by speeding-up random search via adaptively allocating some predefined resources. The resources include the training iterations, the number of training data used, etc. At the same time, there are also work adaptively selecting both the fidelity of the approximation and the point under the Bayesian optimization framework. By extrapolating the performances on the full data trained to completion through smaller training

iterations and/or subset of the training data, the novel Bayesian optimization algorithms can save the wall-clock time dramatically, examples include [31, 32, 35, 72]. In this chapter, we explore the idea of using the knowledge gradient to adaptively select the fidelity of the approximation and the configuration to evaluate simultaneously. The method can inherit the parallelization nature proposed in Chapter 2.

5.2 Continuous-Fidelity Knowledge Gradient

Without loss of generality, we would like to solve

$$\min_{x \in \mathbb{A}} f(x)$$

where x is the parameters that we would like to optimize. Now we have access to a function $g(x, y) : \mathbb{A} \times [0, 1]^m \to \mathbb{R}$ where $f(x) := g(x, 1_m)$ and $y \in [0, 1]^m$ denotes the m fidelity-control parameters. We can evaluate $g(x, y)$ at a given x for a given fidelity y.

The above mathematical formulation models the settings where we have access to some low-fidelity but cheap approximations. For example, $f(x)$ can be the validation loss for a machine learning algorithm trained on the full training data until convergence, x are the hyperparameters that we would like to choose to minimize the validation error. Human experts can often speed up the hyperparameters evaluation by training the algorithm on a much smaller data set and/or running the training procedure with much less iterations, which is modeled by the function $g(x, y)$. In this example, y belongs to a two-dimensional hypercube $[0, 1]^2$, the value of $g(x, y_1, y_2)$ denotes the loss on the validation set when training the algorithm with y_1 of the training data and y_2 of the predefined training length (such as 500 epochs).

We jointly model the function $g(x, y)$ via a GP with a mean function μ^0 and a kernel function K^0. After n function evaluations. we have $g(x, y) \sim \mathrm{GP}(\mu^n, K^n)$.

If we were to stop now at the iteration n, the expected quality of the solution was $\min_{x\in\mathbb{A}} \mu^n(x, 1_m)$. If we were given an additional sample x for the fidelity y, then the expected quality became $\min_{x\in\mathbb{A}} \mu^{n+1}(x, 1_m)$. One notes that $\min_{x\in\mathbb{A}} \mu^{n+1}(x, 1_m)$ is random because it depends on the outcome of the observed value at x with the fidelity y. The information gain by sampling at x with the fidelity y is the difference of two expected solution qualities $\min_{x\in\mathbb{A}} \mu^n(x, 1_m) - \min_{x\in\mathbb{A}} \mu^{n+1}(x, 1_m)$. To trade-off the information gain vs. the cost of sampling at some higher fidelities, continuous-fidelity knowledge gradient (cfKG) is defined as the *information gain per unit cost*, i.e.

$$\textit{cf-KG}(z) = \mathrm{E}_n \left[\frac{\min_{x\in\mathbb{A}} \mu^n(x, 1_m) - \min_{x\in\mathbb{A}} \mu^{n+1}(x, 1_m)}{\mathrm{cost}(z)} \Big| z \right]$$

where $z := (x, y)$. We can decide which fidelity to select and where to sample by maximizing the cfKG factor

$$\max_{z\in\mathbb{A}\times[0,1]^m} \textit{cf-KG}(z).$$

Preliminary numerical experiments. We test the performances of cfKG with KG. We add 1 or 2 additional dimensions to the three synthetic functions: Branin, 3-d Hartmann and Rosenbrock. These new dimensions represent the fidelity con-

trols. Specifically, we define

$$\begin{aligned}
\text{modified-Branin}(x,y) &= \left(x_2 - \left(\frac{5.1}{4\pi^2} - 0.01 * (1-y_1)\right) x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 \\
&\quad +10 * \left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10 \\
\text{modified-Hartmann}(x,y) &= (\alpha_1 - 0.1 * (1-y_1)) \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{1j})^2\right) \\
&\quad + \sum_{i=2}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right) \\
\text{modified-Rosenbrock}(x,y) &= \sum_{i=1}^{2} \left[100 * (x_{i+1} - x_i^2 + 0.001 * (1-y_1))^2\right. \\
&\quad \left. +(x_i - 1 + 0.01 * (1-y_2))^2\right].
\end{aligned}$$

The cost function is defined as $\text{cost}(z) = \prod_{i=1}^{m}(0.05 + y_i)$. The numerical results are summarized in Fig. 5.1. One can see that cfKG achieves much lower simple regret with the same cost compared to KG.

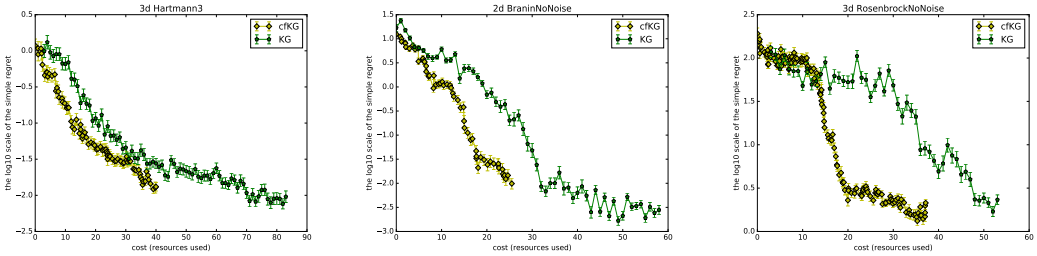

Figure 5.1: The comparisons between KG and continuous-fidelity KG on three synthetic functions: Branin, 3-d Hartmann, and Rosenbrock. The fidelity space is one dimensional for Branin and Hartmann3 and two dimensional for Rosenbrock.

5.3 Summary

In this chapter, we develop a novel BO algorithm, continuous-fidelity knowledge gradient, when there exist some low-fidelity but cheap approximations, and the fidelity is controlled by some continuous parameters. The novel method can out-

perform conventional BO methods which does not exploit these approximations on several synthetic test functions.

CHAPTER 6

CONCLUSION

In this thesis we have generalize the concept of the knowledge gradient to develop novel Bayesian optimization methods: (i) *q-KG* for batch Bayesian optimization; (ii) *d-KG* for Bayesian optimization with gradients; (iii) *cf-KG* for continuous-fidelity Bayesian optimization. We have shown that the proposed methods perform well in both synthetic test functions and/or when tuning machine learning hyperparameters, and enjoyed nice theoretical properties: they are one step Bayes-optimal, and asymptotically consistent.

The developed methods have broad applicability in fields such as nature science, engineering design, and machine learning, and we sincerely hope that they will make the knowledge gradient methods widely used in Bayesian optimization. In the mean time, there are ongoing work by the author and other researchers in improving the current methods proposed in this thesis (and the general Bayesian optimization), and we give a short list here.

- Risk-averse Knowledge Gradient: KG generalizes EI such that the final recommended point can be searched over the whole domain (not restricted to the previous sampled points). KG recommends the point $x \in \arg\max_{x \in \mathbb{A}} \mu^N(x)$ after running out of the sampling budget N, however, this recommendation is risk-neutral, which does not account for the uncertainty around the recommended point. On the other end, EI only recommends the point previously sampled, about which we are quite certain. KG performs quite well if the statistical model describes the data reasonably well but may recommend a really bad point if the statistical model is far away from the real data. EI

does not make as quick progress as KG when the statistical model is good but it will not make a worse recommendation even when the statistical model is not that good in the noise-free cases. In the *Risk-averse knowledge gradient*, we will recommend the final point in the region where our uncertainty is within certain threshold τ, i.e.

$$x^* \in \arg \min_{\{\sqrt{K^N(x,x)} \leq \tau\} \cap \mathbb{A}} \mu^N(x).$$

By Lagrange relaxation, it is "close to"

$$x^* \in \arg \min_{x \in \mathbb{A}} \mu^N(x) + \beta \sqrt{K^N(x,x)}.$$

for some $\beta \geq 0$ which is related to τ. Then the one-step Bayes-optimal way to value the sampling point $x^{n+1} = z$ is

$$\text{raKG}(z) = \min_{x \in \mathbb{A}} \left(\mu^n(x) + \beta \sqrt{K^n(x,x)} \right) - \text{E} \left[\min_{x \in \mathbb{A}} \left(\mu^{n+1}(x) + \beta \sqrt{K^{n+1}(x,x)} \right) \middle| x^{n+1} = z \right].$$

We will recommend to sample point z by solving

$$\max_{z \in \mathbb{A}} \text{raKG}(z).$$

The disadvantage of this method compared to both EI and KG is that β requires tuning, but $\beta = 1$ is a reasonable value to start with.

- Scalable Bayesian Optimization: The $\mathcal{O}(n^3)$ complexity of the GP inference prevents the application of BO to ten of thousands of function evaluations. Some researchers [67, 70] suggest replacing GP with a flexible parametric model, such as neural networks. However, the parametric model is easy to suffer from inaccurate uncertainty quantification [78]. One could combine methods in this thesis with flexible deep projections [82], and recent advances in scalable Gaussian processes for $\mathcal{O}(n)$ training and $\mathcal{O}(1)$ test time predictions [81, 83]. These steps would help make Bayesian optimization

applicable to a much wider range of problems, wherever standard gradient based optimizers are used – even when we have analytic objective functions that are *not* expensive to evaluate – while retaining faster convergence and robustness to multimodality.

- High-dimensional Bayesian Optimization: Though Bayesian optimization has achieved tremendous success in low dimensional blackbox optimization problems (with dimension 10), its scaling to high dimension (100) is of fundamental challenging. [79] suggests optimizing the acquisition functions in a random embedded subspace. [33] suggests the use of additive kernels to tackle the high dimensional problems. Once could consider combining these advances with the techniques in this thesis.
- Convergence Rate of KG: Recently, [2] proves that KG converges to the global optima when the budget goes to infinity if the function is sampled from the running GP with known hyperparameters. However, the rate of convergence is still an open problem.

In addition to developing the methods, the author, developed an open source implementation at https://github.com/wujian16/qKG, which inherited the open source software package for Bayesian optimization, "Metrics Optimization Engine" [75]. The implementation offers the interface in Python, and can be useful to both researchers who would like to extend the methods in this thesis and practitioners who would like to apply the methods here to their own problems.

The author sincerely hope that the work in this thesis can make knowledge gradient methods widely known and applied in Bayesian optimization, and have impact on solving problems in nature science, engineering design, and machine learning.

APPENDIX A

SUPPLEMENTARY MATERIALS FOR CHAPTER 2

A.1 Asynchronous q-KG Optimization

The (A.1.1) corresponds to the synchronous q-KG optimization, in which we wait for all q points from our previous batch to finish before searching for a new batch of q points. However, in some applications, we may wish to generate a new batch of points to evaluate next while $p(< q)$ points are still being evaluated, before we have their values. This is common in training machine learning algorithms, where different machine learning models do not necessarily finish at the same time.

$$\max_{\boldsymbol{z}^{(1:q)} \subset \mathbb{A}} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}). \tag{A.1.1}$$

We can generalize (A.1.1) to the asynchronous q-KG optimization. Given that p points are still under evaluation, now we would like to recommend a batch of $q - p$ points to evaluate. As we did for the synchronous q-KG optimization above, now we estimate the gradient of q-KG of the combined q points only with respect to the $q - p$ points that we need to recommend. Then we proceed the same way via gradient-based algorithms.

A.2 Speed-up analysis

Next, we compare q-KG at different levels of parallelism against the fully sequential KG algorithm. We test the algorithms with different batch sizes on two noisy

synthetic functions Branin2 and Hartmann6, whose standard deviation of the noise is $\sigma = 0.5$. From the results, our parallel knowledge gradient method does provide a speed-up as q goes up.

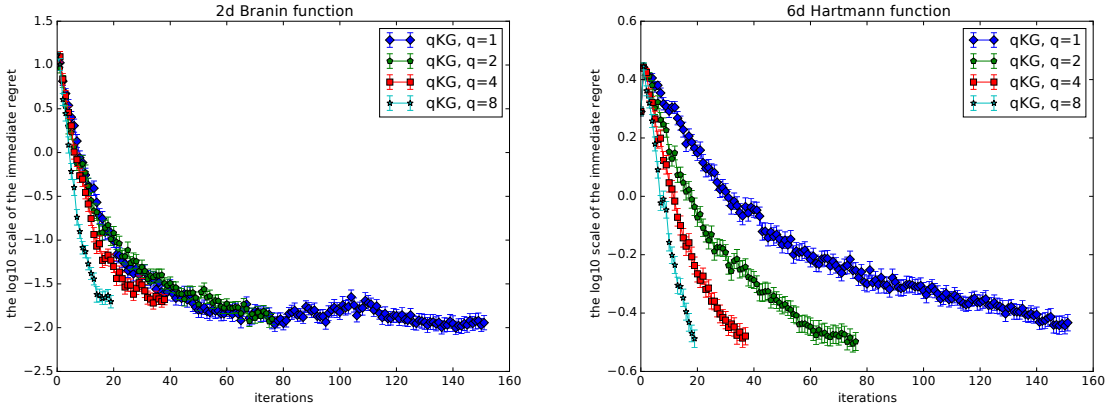

Figure A.1: The performances of *q-KG* with different batch sizes. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of iterations. Iteration 0 is the initial designs. For each iteration later, we evaluate q points recommended by the *q-KG* algorithm.

A.3 The unbiasedness of the stochastic gradient estimator

Recall that in Section 5 of the main document, we have expressed the *q-KG* factor as follows,

$$q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) \quad = \quad \mathbb{E}\left(g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q)\right) \tag{A.3.1}$$

where the expectation is taken over Z_q and

$$\begin{aligned} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q) &= \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) - \min_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) Z_q\right), \\ \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) &= K^{(n)}(x, \boldsymbol{z}^{(1:q)})(D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1}. \end{aligned}$$

The main purpose of this section is to prove the following proposition.

Proposition 5. *When $\mathbb{A}$ is finite, under the condition that μ and K are continuous differentiable,*

$$\frac{\partial}{\partial z_{ij}} q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A})\Big|_{\boldsymbol{z}^{(1:q)}=\theta^{(1:q)}} = \mathbb{E}\left(\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q)\right)\Big|_{\boldsymbol{z}^{(1:q)}=\theta^{(1:q)}}, \tag{A.3.2}$$

where $1 \leq i \leq q$, $1 \leq j \leq d$, z_{ij} is the jth dimension of the ith point in $\boldsymbol{z}^{(1:q)}$ and $\theta^{(1:q)} \in$ the interior of $\mathbb{A}^q$.

Without loss of generality, we assume that (1) i and j are fixed in advance and (2) $\mathbb{A} = [0,1]^d$, we would like to prove that (A.3.2) is correct. Before proceeding, we define one more notation $f_{\mathbb{A},Z_q}(z_{ij}) := g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q)$ where $\boldsymbol{z}^{(1:q)}$ equals to $\theta^{(1:q)}$ component-wise except for z_{ij}. To prove it, we cite Theorem 1 in [40], which requires three conditions to make (A.3.2) valid: there exists an open neighborhood $\Theta = (0,1)$ of θ_{ij} where θ_{ij} is the jth dimension of ith point in $\theta^{(1:q)}$ such that (i) $f_{\mathbb{A},Z_q}(z_{ij})$ is continuous in Θ for any fixed $\mathbb{A}$ and Z_q, (ii) $f_{\mathbb{A},Z_q}(z_{ij})$ is differentiable except on a denumerable set in Θ for any given $\mathbb{A}$ and Z_q, (iii) the derivative of $f_{\mathbb{A},Z_q}(z_{ij})$ (when it exists) is uniformly bounded by $\Gamma(Z_q)$ for all $z_{ij} \in \Theta$, and the expectation of $\Gamma(Z_q)$ is finite.

A.3.1 The proof of condition (i)

Under the condition that the mean function μ and the kernel function K are continuous differentiable, we see that for any given x, $\tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)})$ is continuous differentiable in $\boldsymbol{z}^{(1:q)}$ by the result that the multiplication, the inverse (when the inverse exists) and the Cholesky operators [65] preserve continuous differentiability. When A is finite, we see that $g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q) = \min_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x) -$

$\min_{x\in\mathbb{A}}\left(\boldsymbol{\mu}^{(n)}(x)+\tilde{\sigma}_n(x,\boldsymbol{z}^{(1:q)})Z_q\right)$ is continuous in $\boldsymbol{z}^{(1:q)}$. Then $f_{\mathbb{A},Z_q}(z_{ij})$ is also continuous in z_{ij} by the definition of the function $f_{\mathbb{A},Z_q}(z_{ij})$.

A.3.2 The proof of condition (ii)

By the expression that $f_{\mathbb{A},Z_q}(z_{ij})=\min_{x\in\mathbb{A}}\boldsymbol{\mu}^{(n)}(x)-\min_{x\in\mathbb{A}}\left(\boldsymbol{\mu}^{(n)}(x)+\tilde{\sigma}_n(x,\boldsymbol{z}^{(1:q)})Z_q\right)$, if both $\text{argmin}_{x\in\mathbb{A}}\boldsymbol{\mu}^{(n)}(x)$ and $\text{argmin}_{x\in\mathbb{A}}\left(\boldsymbol{\mu}^{(n)}(x)+\tilde{\sigma}_n(x,\boldsymbol{z}^{(1:q)})Z_q\right)$ are unique, then $f_{\mathbb{A},Z_q}(z_{ij})$ is differentiable at z_{ij}. We define $D(\mathbb{A})\subset\Theta$ to be the set that $f_{\mathbb{A},Z_q}(z_{ij})$ is not differentiable, then we see that

$$\begin{aligned}D(\mathbb{A}) \subset\ & \cup_{x,x'\in\mathbb{A}}\left\{z_{ij}\in\Theta:\boldsymbol{\mu}^{(n)}(x)=\boldsymbol{\mu}^{(n)}(x'),\frac{d\boldsymbol{\mu}^{(n)}(x)}{dz_{ij}}\neq\frac{d\boldsymbol{\mu}^{(n)}(x')}{dz_{ij}}\right\}\cup\\ & \cup_{x,x'\in\mathbb{A}}\left\{z_{ij}\in\Theta:h_x(z_{ij})=h_{x'}(z_{ij}),\frac{dh_x(z_{ij})}{dz_{ij}}\neq\frac{dh_{x'}(z_{ij})}{dz_{ij}}\right\}\end{aligned}$$

where $h_x(z_{ij}) := \boldsymbol{\mu}^{(n)}(x)+\tilde{\sigma}_n(x,\boldsymbol{z}^{(1:q)})Z_q$. $\boldsymbol{\mu}^{(n)}(x)\left(\boldsymbol{\mu}^{(n)}(x')\right)$ depend on z_{ij} if $x=z_i$ ($x'=z_i$) where z_i is the ith point of $\boldsymbol{z}^{(1:q)}$. As $\mathbb{A}$ is finite, we only need to show that $\left\{z_{ij}\in\Theta:\boldsymbol{\mu}^{(n)}(x)=\boldsymbol{\mu}^{(n)}(x'),\frac{d\boldsymbol{\mu}^{(n)}(x)}{dz_{ij}}\neq\frac{d\boldsymbol{\mu}^{(n)}(x')}{dz_{ij}}\right\}$ and $\left\{z_{ij}\in\Theta:h_x(z_{ij})=h_{x'}(z_{ij}),\frac{dh_x(z_{ij})}{dz_{ij}}\neq\frac{dh_{x'}(z_{ij})}{dz_{ij}}\right\}$ is denumerable.

Defining $\eta(z_{ij}) := h_{x_1}(z_{ij})-h_{x_2}(z_{ij})$ on Θ, one can see that $\eta(z_{ij})$ is continuous differentiable on Θ. We would like to show that $E:=\left\{z_{ij}\in\Theta:\eta(z_{ij})=0,\frac{d\eta(z_{ij})}{dz_{ij}}\neq 0\right\}$ is denumerable. To prove it, we will show that E contains only isolated points. Then one can use a theorem in real analysis: any set of isolated points in $\mathbb{R}$ is denumerable (see the proof of statement 4.2.25 on page 165 in [73]). To prove that E only contains isolated points, we use the definition of an isolated point: $y\in E$ is an isolated point of E if and only if $x\in E$ is not a limit point of E. We will prove by contradiction, suppose that $y\in E$ is a limit point of E, then it means that there exists a sequence of points $y_1,y_2,\cdots$ all belong to E such that $\lim_{n\to\infty}y_n=z_{ij}$. However, by the definition of derivative and

$\eta(y_n) = \eta(z_{ij}) = 0$, $0 \neq \frac{d\eta(y)}{dy}\big|_{y=z_{ij}} = \lim_{n\to\infty} \frac{\eta(y_n)-\eta(z_{ij})}{y_n - z_{ij}} = \lim_{n\to\infty} 0 = 0$, a contradiction. So we conclude that E only contains isolated points, so is denumerable.

Defining $\delta(z_{ij}) := \boldsymbol{\mu}^{(n)}(x_1) - \boldsymbol{\mu}^{(n)}(x_2)$ on Θ, $\delta(z_{ij})$ is also continuous differentiable on Θ, then one can similarly prove that $\left\{ z_{ij} \in \Theta : \delta(z_{ij}) = 0, \frac{d\delta(z_{ij})}{dz_{ij}} \neq 0 \right\}$ is denumerable.

A.3.3 The proof of condition (iii)

Recall that from Section 5 of the main document,

$$\begin{aligned}
\frac{d}{dz_{ij}} f(z_{ij}, \mathbb{A}, Z_q) &= \frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q) \\
&= \frac{\partial}{\partial z_{ij}} \boldsymbol{\mu}^{(n)}(x^*(\text{before})) - \frac{\partial}{\partial z_{ij}} \boldsymbol{\mu}^{(n)}(x^*(\text{after})) \\
&\quad - \frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(\boldsymbol{z}^{(1:q)}, x^*(\text{after})) Z_q,
\end{aligned}$$

where $x^*(\text{before}) = \text{argmin}_{x\in\mathbb{A}} \boldsymbol{\mu}^{(n)}(x)$, $x^*(\text{after}) = \text{argmin}_{x\in\mathbb{A}} \left(\boldsymbol{\mu}^{(n)}(x) + \tilde{\sigma}_n(x, \boldsymbol{z}^{(1:q)}) Z_q \right)$, and

$$\begin{aligned}
\frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(\boldsymbol{z}^{(1:q)}, x^*(\text{after})) &= \left(\frac{\partial}{\partial z_{ij}} K^{(n)}(x^*(\text{after}), \boldsymbol{z}^{(1:q)}) \right) (D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\
&\quad - D^{(n)}(x^*(\text{after}), \boldsymbol{z}^{(1:q)}) (D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1} \\
&\quad \left(\frac{\partial}{\partial z_{ij}} D^{(n)}(\boldsymbol{z}^{(1:q)})^T \right) (D^{(n)}(\boldsymbol{z}^{(1:q)})^T)^{-1}.
\end{aligned}$$

We can calculate the $\frac{\partial}{\partial z_{ij}} \boldsymbol{\mu}^{(n)}(x)$ as follows

$$\frac{\partial}{\partial z_{ij}} \boldsymbol{\mu}^{(n)}(x) = \begin{cases} \frac{\partial}{\partial z_{ij}} \boldsymbol{\mu}^{(n)}(z_i) & \text{if } x = z_i, \text{ i.e. the } i\text{th point of } \boldsymbol{z}^{(1:q)} \\ 0 & \text{otherwise.} \end{cases}$$

Using the fact that μ is continuously differentiable and $\mathbb{A}$ is compact, then $\frac{\partial}{\partial z_{ij}} \boldsymbol{\mu}^{(n)}(x)$ is bounded by some $B > 0$. By the result that $\frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(\boldsymbol{z}^{(1:q)}, x^*(\text{after}))$

is continous, it is bounded by a vector $0 \leq \Lambda < \infty$ as $\mathbb{A}$ is compact. Then $\left|\frac{d}{dz_{ij}} f(z_{ij}, \mathbb{A}, Z_q)\right| \leq 2B + \sum_{i=1}^{q} \Lambda_i |z_i|$ where $Z_q = (z_1, \cdots, z_q)^T$. And $\mathbb{E}\left(\sum_{i=1}^{q} \Lambda_i |z_i|\right) = \sqrt{2/\pi} \sum_{i=1}^{q} \Lambda_i < \infty$.

A.4 The convergence of stochastic gradient ascent

In this section, we will prove that SGA converges to a stationary point. We follow the same idea of proving the Theorem 2 in [76].

First, it requires the step size γ_t satisfying $\gamma_t \to 0$ as $t \to \infty$, $\sum_{t=0}^{\infty} \gamma_t = \infty$ and $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. Second, it requires the second moment of the gradient estimator is finite. In the above section 1.3, we have show that $|\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q)| \leq 2B + \sum_{i=1}^{q} \Lambda_i |z_i|$, then $\mathbb{E}(\frac{\partial}{\partial z_{ij}} g(\boldsymbol{z}^{(1:q)}, \mathbb{A}, Z_q))^2 \leq 4B^2 + \sum_{i=1}^{q} \Lambda_i^2 + 4B\sqrt{2/\pi} \sum_{i=1}^{q} \Lambda_i + 2\sum_{i \neq j} \Lambda_i \Lambda_j < \infty$.

APPENDIX B

SUPPLEMENTARY MATERIALS FOR CHAPTER 4

B.1 The Posterior Distribution of the Multi-Output GP

Suppose that we have sampled f at n points $X := \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}\}$ so far and observed $y^{(1:n)}$, where each observation consists of the function value and the gradient at $\boldsymbol{x}^{(i)}$. Then the posterior distribution is a multi-output Gaussian process with mean function $\tilde{\mu}^n(\cdot)$ and kernel function $\tilde{K}^n(\cdot,\cdot)$, where

$$\begin{aligned}\tilde{\mu}^{(n)}(x) &= \tilde{\mu}(x) + \tilde{K}(x, X)\left(\tilde{K}(X, X) + \operatorname{diag}\{\sigma^2(\boldsymbol{x}^{(1)}), \cdots, \sigma^2(\boldsymbol{x}^{(n)})\}\right)^{-1}(y^{(1:n)} - \tilde{\mu}(X)),\\ \tilde{K}^{(n)}(x_1, x_2) &= \tilde{K}(x_1, x_2) - \tilde{K}(x_1, X)\left(\tilde{K}(X, X) + \operatorname{diag}\{\sigma^2(\boldsymbol{x}^{(1)}), \cdots, \sigma^2(\boldsymbol{x}^{(n)})\}\right)^{-1}\tilde{K}(X, x_2).\end{aligned} \tag{B.1.1}$$

The rows and columns in Eq. (B.1.1) corresponding to partial derivatives (or function values) that were not observed are to be omitted.

B.2 The Computation of *d-KG* and its Gradient: Additional Details

In this section we show additional details in Sect. 4.3.3 of the main document: how the *d-KG*$(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A})$ factor and its gradient can be computed "exactly". It is well-know that if $\tilde{\mu}^{(n)}$ and $\tilde{K}^{(n)}$ are the mean and kernel function respectively of the posterior of $(f(x), \nabla f(x))^T$ after evaluating n points, then $(f(x), \theta^T \nabla f(x)))^T$ follows a bivariate Gaussian process with mean function $\hat{\mu}^{(n)}$ and kernel function

$\hat{K}^{(n)}$ as follows

$$\hat{\mu}^{(n)}(x) = \begin{pmatrix} 1 & 0_{1\times d} \\ 0 & \theta^T \end{pmatrix} \tilde{\mu}^{(n)}(x) \text{ and } \hat{K}^{(n)}(x^1, x^2) = \begin{pmatrix} 1 & 0_{1\times d} \\ 0 & \theta^T \end{pmatrix} \tilde{K}^{(n)}(x^1, x^2) \begin{pmatrix} 1 & 0_{1\times d} \\ 0 & \theta^T \end{pmatrix}^T.$$

Analogously, the $(y(x), \theta^T \nabla y(x))^T$ is also subject to noise,

$$\left(y(x), \theta^T \nabla y(x)\right)^T \Big| \, f(x), \theta^T \nabla f(x) \sim \mathcal{N}\left(\left(f(x), \theta^T \nabla f(x)\right)^T, \text{diag}(\hat{\sigma}^2(x))\right),$$

where $\hat{\sigma}^2(x) = \begin{pmatrix} 1 & 0_{1\times d} \\ 0 & \theta^T \end{pmatrix} \sigma^2(x)$. Conditioned on $\boldsymbol{z}^{(1:q)}$ and the knowledge after n evaluations, we have $(y, \theta^T \nabla y)(\boldsymbol{z}^{(1:q)})$ is normally distributed with mean $\hat{\mu}^{(n)}(\boldsymbol{z}^{(1:q)})$ and covariance matrix $\hat{K}^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) + \text{diag}\{\hat{\sigma}^2(\boldsymbol{z}^{(1)}), \cdots, \hat{\sigma}^2(\boldsymbol{z}^{(q)})\}$ where the multivariate function $(y, \theta^T \nabla y) : \mathbb{R}^d \to \mathbb{R}^{d+1}$ maps the sample to its function and gradient observation.

Following [84], we express $\hat{\mu}^{(n+q)}(x)$ as

$$\begin{aligned} \hat{\mu}^{(n+q)}(x) &= \hat{\mu}^{(n)}(x) + \hat{K}^{(n)}(x, \boldsymbol{z}^{(1:q)}) \Big(\hat{K}^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) \\ &\quad + \text{diag}\{\hat{\sigma}^2(\boldsymbol{z}^{(1)}), \cdots, \hat{\sigma}^2(\boldsymbol{z}^{(q)})\} \Big)^{-1} \left((y, \theta^T \nabla y)(\boldsymbol{z}^{(1:q)}) - \hat{\mu}^{(n)}(\boldsymbol{z}^{(1:q)}) \right). \end{aligned}$$

Thus, we can rewrite $\hat{\mu}^{(n+q)}(x)$ as

$$\hat{\mu}^{(n+q)}(x) = \hat{\mu}^{(n)}(x) + \hat{\sigma}^{(n)}(x, \boldsymbol{z}^{(1:q)}) Z_{2q}, \tag{B.2.1}$$

where Z_{2q} is a $2q$-dimensional standard normal vector and

$$\hat{\sigma}^{(n)}(x, \boldsymbol{z}^{(1:q)}) = \hat{K}^{(n)}(x, \boldsymbol{z}^{(1:q)}) \left(\hat{D}^{(n)}(\boldsymbol{z}^{(1:q)})^T \right)^{-1}. \tag{B.2.2}$$

Here $\hat{D}^{(n)}(\boldsymbol{z}^{(1:q)})$ is the Cholesky factor of the covariance matrix $\hat{K}^{(n)}(\boldsymbol{z}^{(1:q)}, \boldsymbol{z}^{(1:q)}) + \text{diag}\{\hat{\sigma}^2(\boldsymbol{z}^{(1)}), \cdots, \hat{\sigma}^2(\boldsymbol{z}^{(q)})\}$. Now we can follow Sect. 4.3.3 of the main document to compute the d-$KG(\boldsymbol{z}^{(1:q)}, \theta, \mathbb{A})$ factor and its gradient.

B.3 Additional Experimental Results

In this section, we provide a detailed description of all the synthetic test results, and depict additional plots. We evaluate all methods on six test functions chosen from [3]. In order to demonstrate the ability to benefit from *noisy derivative information*, we sample additive normally distributed noise with zero mean and standard deviation $\sigma = 0.5$ for both the objective function and its partial derivatives. Note that σ is not known to the algorithms but has to be estimated from observations. Moreover, we investigate how the performance of the algorithms is affected if partial derivatives are not given for all parameters. We also experiment with two different batch sizes: we use a batch size $q = 4$ for the Branin, Rosenbrock, and Ackley functions; otherwise, we use a batch size $q = 8$. The experimental results are summarized in Fig. 4.2 of the main text and Fig. B.1 here.

Functions with Full Gradient Information. For 2d Branin on domain $[-15, 15]^2$, 5d Ackley on $[-2, 2]^5$, 6d Hartmann function on $[0, 1]^6$, we assume that the full gradient is available.

Looking at the results for the Branin function (cp. Fig. 4.2 in the main text), *d-KG* outperforms its competitors after 40 function evaluations and obtains the best solution overall (within the limit of function evaluations). BFGS makes faster progress than the Bayesian optimization methods during the first 20 evaluations, but subsequently stalls and fails to obtain a competitive solution. On the Ackley function *d-EI* makes fast progress during the first 50 evaluations but also fails to make any subsequent progress. Conversely, *d-KG* requires about 50 evaluations to improve on the performance of *d-EI*; *d-KG* exhibits the best overall performance

again. For the Hartmann function *d-KG* clearly dominates its competitors over all function evaluations.

Functions with Incomplete Derivative Information. For the 3d Rosenbrock function on $[-2, 2]^3$ we only provide a noisy observation of the third partial derivative. Both *EI* and *d-EI* get stuck early. *d-KG* on the other hand finds a near optimal solution after about 50 function evaluations; *q-KG* catches up after about 75 evaluations and has a comparable performance afterwards. The 4d Levy benchmark on $[-10, 10]^4$, where the fourth partial derivative is observable with noise, shows a different ordering of the algorithms: here *EI* has the best performance, beating even its formulation that utilizes derivative information. A possible explanation could be that the smoothness and regularized shape of the function surface benefits this acquisition criterion. For the 8d Cosine mixture function on $[-1, 1]^8$ we provide two noisy partial derivatives. *d-KG* and UCB with derivatives perform better than EI-type criterion, and achieve the best performances, with *d-KG* beating UCB with derivatives slightly.

Summing up, we see that *d-KG* successfully exploits noisy derivative information and has the best overall performance.

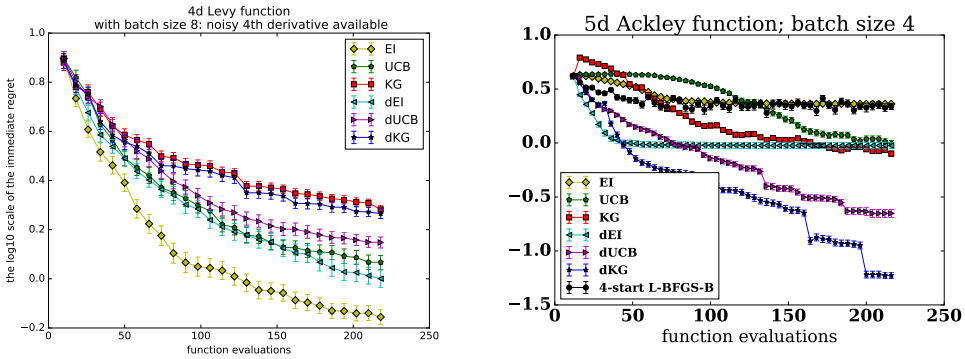

Figure B.1: The average performance of 100 replications (the log10 of the immediate regret vs. the number of function evaluations) for the Levy and Ackley functions. For the Ackley function, we assume that a noisy observation of the full gradient is available. On the Levy function only 4th partial derivative can be observed (with noise). *d-KG* performs significantly better than its competitors for all benchmarks except the Levy function.

B.4 Proof of Proposition 3 and Proposition 4

Proof of Proposition 3. Recall that we start with the same posterior $\tilde{\mu}^{(n)}$, then

$$
\begin{aligned}
& d\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}) \\
= \; & \min_{x\in\mathbb{A}} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n\left[\min_{x\in\mathbb{A}} \mathbb{E}_n\left[y(x)|y(\boldsymbol{z}^{(1:q)}), \nabla y(\boldsymbol{z}^{(1:q)})\right] \,\middle|\, \boldsymbol{z}^{(1:q)}\right], \\
= \; & \min_{x\in\mathbb{A}} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n\left[\mathbb{E}_n\left[\min_{x\in\mathbb{A}} \mathbb{E}_n\left[y(x)|y(\boldsymbol{z}^{(1:q)}), \nabla y(\boldsymbol{z}^{(1:q)})\right] |y_1(\boldsymbol{z}^{(1:q)})\right] \,\middle|\, \boldsymbol{z}^{(1:q)}\right], \\
\geq \; & \min_{x\in\mathbb{A}} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n\left[\min_{x\in\mathbb{A}} \mathbb{E}_n\left[\mathbb{E}_n\left[y(x)|y(\boldsymbol{z}^{(1:q)}), \nabla y(\boldsymbol{z}^{(1:q)})\right] |y(\boldsymbol{z}^{(1:q)})\right] \,\middle|\, \boldsymbol{z}^{(1:q)}\right], \\
= \; & \min_{x\in\mathbb{A}} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n\left[\min_{x\in\mathbb{A}} \mathbb{E}_n\left[y(x)|y(\boldsymbol{z}^{(1:q)})\right] \,\middle|\right], \\
= \; & q\text{-}KG(\boldsymbol{z}^{(1:q)}, \mathbb{A}),
\end{aligned}
\tag{B.4.1}
$$

where recall that $y(x)$ is the observed function value at x, and $\nabla y(x)$ are the d derivative observations at x accordingly. The inequality above holds due to Jensen's inequality. □

Next we analyze the Bayesian optimization problem under the dynamic programming (DP) framework and show that *d-KG* is one-step Bayes-optimal.

Proof of Proposition 4. Suppose that we are given N iteration budgets, our goal is to choose sampling decisions $(\{z^i, 1 \le i \le Nq\}$ and implementation decision z^{Nq+1} that minimizes $f(z^{Nq+1})$. We assume that $(f(x), \nabla f(x))$ is drawn from the prior $\mathcal{GP}(\tilde{\mu}, \tilde{K})$, then $(f(x), \nabla f(x))$ also follows the posterior process $\mathcal{GP}(\tilde{\mu}^{(Nq)}, \tilde{K}^{(Nq)})$ after N iterations, so we have $\mathbb{E}_{Nq}(f(z^{Nq+1})) = \tilde{\mu}_1^{(Nq)}(z^{Nq+1})$. Thus, letting Π be the set of feasible policies π, we can formulate our problem as follows

$$\inf_{\pi \in \Pi} \mathbb{E}^{\pi} \left[\min_{x \in \mathbb{A}} \tilde{\mu}_1^{(Nq)}(x) \right].$$

We analyze this problem under the DP framework. We define our state space as $S^n := (\tilde{\mu}^{(nq)}, \tilde{K}^{(nq)})$ after iteration n as it completely characterizes our belief on f. Under the DP framework, we need to define the value function V^n as follows

$$V^n(s) := \inf_{\pi \in \Pi} \mathbb{E}^{\pi} \left[\min_{x \in \mathbb{A}} \tilde{\mu}_1^{(Nq)}(x) \middle| S^n = s \right] \tag{B.4.2}$$

for every $s = (\mu, K)$. The bellman equation tells us that the value function can be written recursively by

$$V^n(s) \;=\; \min_{\boldsymbol{z} \in \mathbb{A}^q} Q^n(s, \boldsymbol{z})$$

where

$$Q^n(s, \boldsymbol{z}) \;=\; \mathbb{E}\left[V^{n+1}(S^{n+1}) | S^n = s, \boldsymbol{z}^{((nq+1):(n+1)q)} = \boldsymbol{z} \right]$$

At the same time, we also know that any policy π^* whose decision satisfy

$$Z^{\pi^*, n}(s) \in \operatorname{argmin}_{\boldsymbol{z} \in \mathbb{A}^q} Q^n(s, \boldsymbol{z}) \tag{B.4.3}$$

is optimal. If we were to stop at iteration $n+1$, then $V^{n+1}(S^{n+1}) = \min_{x \in \mathbb{A}} \tilde{\mu}_1^{((n+1)q)}(x)$ and (B.4.3) reduces to

$$\begin{aligned} Z^{\pi^*, n}(s) \;&\in\; \operatorname{argmin}_{\boldsymbol{z} \in \mathbb{A}^q} \mathbb{E}\left[\min_{x \in \mathbb{A}} \tilde{\mu}_1^{((n+1)q)}(x) \;\middle|\; S^n = s, \boldsymbol{z}^{((nq+1):(n+1)q)} = \boldsymbol{z} \right] \\ &=\; \operatorname{argmax}_{\boldsymbol{z} \in \mathbb{A}^q} \min_{x \in \mathbb{A}} e_1^T \tilde{\mu}^{(nq)}(x) - \mathbb{E}\left[\min_{x \in \mathbb{A}} \tilde{\mu}_1^{((n+1)q)}(x) \;\middle|\; S^n = s, \boldsymbol{z}^{((nq+1):(n+1)q)} = \boldsymbol{z} \right], \end{aligned}$$

which is exactly the *d-KG* algorithm. This proves that *d-KG* is one-step Bayes-optimal. □

B.5 Proof of Theorem 2

At the beginning of this section, we will state two results concerning the benefits of additional samples, which will be useful in the latter proofs. Recall that we define the value function in Eq. (B.4.2). Similarly, we can define the value function for a specific policy π as

$$V^{\pi,n}(s) := \mathbb{E}^{\pi}\left[\min_{x\in\mathbb{A}} \tilde{\mu}_1^{(Nq)}(x)|S^n = s\right]. \tag{B.5.1}$$

Since we are varying the number of iterations N, we define $V^0(s;N)$ as the optimal value function when the number of iteration budgets is N. Additionally, we define $V(s;\infty) := \lim_{N\to\infty} V^0(s;N)$. Similarly, we define $V^{0,\pi}(s;N)$ and $V^{\pi}(s;\infty)$ for a specific policy π. Policy π is asymptotically consistent if $V^{\pi}(s;\infty) = V(s;\infty)$. We have the following result for any stationary policy π.

Lemma 1. *For any stationary policy π and state s, $V^{\pi,n}(s) \leq V^{\pi,n+1}(s)$.*

This lemma states that for any stationary policy, one additional iteration helps on average.

Proof of Lemma 1. We prove by induction on n. When $n = N-1$, by Jensen's inequality,

$$\begin{aligned} V^{\pi,N-1}(s) &= \mathbb{E}^{\pi}\left[\min_{x} \mu_1^{(Nq)}(x) \mid S_{N-1} = s\right] \\ &\leq \min_{x} \mathbb{E}^{\pi}\left[\mu_1^{(Nq)}(x) \mid S_{N-1} = s\right] \\ &= V^{\pi,N}(s). \end{aligned}$$

Then by the induction hypothesis,

$$\begin{aligned} V^{\pi,n}(s) &= \mathbb{E}^{\pi}\left[V^{\pi,n+1}(T(s,\pi))\right] \\ &\leq \mathbb{E}_{s}\left[V^{\pi,n+2}(T(s,\pi))\right] \\ &= V^{\pi,n+1}(s), \end{aligned}$$

where $T(s,\pi)$ is the transition kernel when the state is s under the stationary policy π. We concludes the proof. □

The following lemma is related to the optimal policy. It says that if allowed an extra fixed batch of samples, the optimal policy performs better on average than if no extra samples allowed.

Lemma 2. *For any state s and $\boldsymbol{z} \in \mathbb{A}$, $Q^{n}(s,x) \leq V^{n+1}(s)$.*

As a direct corollary, we have $V^{n}(s) \leq V^{n+1}(s)$ for any state s.

Proof of Lemma 2. The proof of Lemma 2 is quite similar to that of Lemma 1. We omit the details here. □

The lemma below shows that $V(s;\infty)$ is well defined and bounded below.

Lemma 3. *For any state s, $V(s;\infty)$ exists and*

$$V(s;\infty) \geq U(s) := \mathbb{E}\left[\min_{x\in\mathbb{A}} f(x) | S^{0} = s\right]. \tag{B.5.2}$$

Proof of Lemma 3. We will show that $V^{0}(S^{0};N)$ is non-increasing of N and bounded below from $U(S^{0})$. This will imply that $V^{0}(S^{0};\infty)$ exists and is bounded

below from $U(S^0)$. To prove that $V^0(S^0;N)$ is non-increasing of N, we note that

$$\begin{aligned} & V^0(S^0;N) - V^0(S^0;N-1) \\ = \; & V^0(S^0;N) - V^1(S^0;N) \\ \leq \; & 0. \end{aligned}$$

To show that $V^0(S^0;N)$ is bounded below from $U(S^0)$, for every $N \geq 1$ and policy π,

$$\begin{aligned} \mathbb{E}^\pi \left[\min_x \tilde{\mu}_1^{(Nq)}(x) \right] &= \mathbb{E}^\pi \left[\min_x \mathbb{E}_N^\pi \left[f(x) \right] \right] \\ &\geq \mathbb{E}^\pi \left[\mathbb{E}_N^\pi \left[\min_x f(x) \right] \right] \\ &= \mathbb{E}^\pi \left[\min_x f(x) \right] \\ &= \mathbb{E} \left[\min_x f(x) \right] = U(S^0). \end{aligned}$$

Thus we have $V^0(S^0;N) \geq U(S^0)$. Taking the limit $N \to \infty$, we have $V(S^0,\infty) \geq U(S^0)$.

We will now show that $V^\pi(S^0;\infty)$ exists for each stationary policy. The proof is similar as above. We can show that $V^{\pi,0}(S^0;N)$ is non-increasing in N and bounded below from $U(S^0)$. Hence, $V^\pi(S^0;\infty)$ exists. □

A policy is called *stationary* if the decision of the policy only depends on the current state $S^n := (\tilde{\mu}^{(n)}, \tilde{K}^{(n)})$ (not related to which iteration it is after, i.e. n). *d-KG* is stationary. The following lemma is the key idea to prove the asymptotic consistency.

Lemma 4. *If a stationary policy π measures every alternative $x \in \mathbb{A}$ infinitely often almost surely, then π is asymptotically consistent and has value $U(s)$.*

Proof of Lemma 4. We assume that the measurement noise is of finite variance, it

implies that the posterior sequence $\mu^{(Nq)}$ converges to true surface f by vector-version strong law of large numbers if we sample every alternative infinitely often. Thus, $\lim_{N\to\infty}\mu^{(Nq)} = f$ a.s., and $\lim_{N\to\infty}\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x) = \min_{x\in\mathbb{A}} f(x)$ in probability. Next we will show that $\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)$ is uniformly integrable in N, which implies that $\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)$ converges in L_1. For a fixed $K \geq 0$, we have

$$\begin{aligned}
& \mathbb{E}\left[|\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)|1_{\{|\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)|\geq K\}}\right] \\
\leq\ & \mathbb{E}\left[\max_{x\in\mathbb{A}}|\tilde{\mu}_1^{(Nq)}(x)|1_{\{\max_{x\in\mathbb{A}}|\tilde{\mu}_1^{(Nq)}(x)|\geq K\}}\right] \\
=\ & \mathbb{E}\left[\max_{x\in\mathbb{A}}|\mathbb{E}_{Nq}(f(x))|1_{\{\max_{x\in\mathbb{A}}|\mathbb{E}_{Nq}(f(x))|\geq K\}}\right] \\
\leq\ & \mathbb{E}\left[\max_{x\in\mathbb{A}}\mathbb{E}_{Nq}(|f(x)|)1_{\{\max_{x\in\mathbb{A}}\mathbb{E}_{Nq}(|f(x)|)\geq K\}}\right] \\
\leq\ & \mathbb{E}\left[\mathbb{E}_{Nq}(\max_{x\in\mathbb{A}}|f(x)|)1_{\{\mathbb{E}_{Nq}(\max_{x\in\mathbb{A}}|f(x)|)\geq K\}}\right] \\
=\ & \mathbb{E}\left[\mathbb{E}_{Nq}\left(\max_{x\in\mathbb{A}}|f(x)|1_{\{\mathbb{E}_{Nq}(\max_{x\in\mathbb{A}}|f(x)|)\geq K\}}\right)\right] \\
=\ & \mathbb{E}\left[\max_{x\in\mathbb{A}}|f(x)|1_{\{\mathbb{E}_{Nq}(\max_{x\in\mathbb{A}}|f(x)|)\geq K\}}\right].
\end{aligned}$$

Since $\max_{x\in\mathbb{A}}|f(x)|$ is integrable and $P(\max_{x\in\mathbb{A}}|f(x)|) \geq K) \leq \mathbb{E}(\max_x|f(x)|)/K$ is bounded uniformly in N and goes to zeros as K increases to infinity, Given that $\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)$ converges in L_1, we have

$$\begin{aligned}
V^\pi(S^0;\infty) &= \lim_{N\to\infty}\mathbb{E}^\pi[\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)] \\
&= \mathbb{E}^\pi[\lim_{N\to\infty}\min_{x\in\mathbb{A}}\tilde{\mu}_1^{(Nq)}(x)] \\
&= \mathbb{E}^\pi[\min_{x\in\mathbb{A}} f(x)] \\
&= U(S^0).
\end{aligned}$$

So by Lemma 3 in the main document, we concludes that $V^\pi(S^0;\infty) = V(S^0;\infty) = U(S^0)$. □

Then we will show that *d-KG* measures every alternative $x \in \mathbb{A}$ infinitely often

when N goes to infinity, which leads to the proof of Theorem 2.

Proof of Theorem 2. Note that *d-KG* is a stationary policy, we only need to show that *d-KG* algorithm samples every alternative infinitely often if N goes to infinity.

By the similar proof with Lemma A.5 in [13], we can show that S^n converges to a random variable $S^\infty := (\tilde{\mu}^\infty, \tilde{K}^\infty)$ as n increases. By definition,

$$\begin{aligned} & V^N(S^\infty) - Q^{N-1}(S^\infty; x) \\ = \; & \min_x \tilde{\mu}_1^\infty(x) - \mathbb{E}\left[\min_x \left(\tilde{\mu}_1^\infty(x) + e_1^T \tilde{\sigma}^\infty(x, \boldsymbol{z}^{(1:q)}) Z_{q(d+1)}\right)\right] \end{aligned}$$

If we have measured x infinitely often, there will be no uncertainty around $f(x)$ in S^∞, then $V^N(S^\infty) = Q^{N-1}(S^\infty; x)$. If we have not measured x infinitely often, then $V^N(S^\infty) > Q^{N-1}(S^\infty; x)$, i.e. there are benefits measuring x. We define $E = \{x \in \mathbb{A} : \text{the number of times measuring } x < \infty\}$, then for any $x \in E$ and $y \in E^c$, we have $Q^{N-1}(S^\infty; x) < V^N(S^\infty) = Q^{N-1}(S^\infty; y)$. By the definition of *d-KG*, it will measure some $x \in E$, i.e. at least one of x in E is measured infinitely often, a contradiction. □

BIBLIOGRAPHY

[1] M. O. Ahmed, B. Shahriari, and M. Schmidt. Do we need "harmless" bayesian optimization and "first-order" bayesian optimization? BayesOpt, 2016.

[2] J. Bect, F. Bachoc, and D. Ginsbourger. A supermartingale approach to gaussian process based sequential design of experiments. *arXiv preprint arXiv:1608.01118*, 2016.

[3] D. Bingham. Optimization test problems. http://www.sfu.ca/~ssurjano/optimization.html, 2015.

[4] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[5] C. Chevalier and D. Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In *Learning and Intelligent Optimization*, pages 59–69. Springer, 2013.

[6] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[7] J. Dai, M. Miyazawa, and J. Wu. A multi-dimensional srbm: geometric views of its product form stationary distribution. *Queueing Systems*, 78(4):313–335, 2014.

[8] J. Dai, M. Miyazawa, and J. Wu. Decomposable stationary distribution of a multidimensional srbm. *Stochastic Processes and their Applications*, 125(5):1799–1820, 2015.

[9] L. Deng and D. Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[10] T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

[11] E. C. etal. Gpoptimization. https://reine.cmla.ens-cachan.fr/e.contal/gpoptimization, 2015.

[12] A. Forrester, A. Sobester, and A. Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[13] P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009.

[14] P. I. Frazier, W. B. Powell, and S. Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

[15] P. I. Frazier and J. Wang. Bayesian optimization for materials design. In *Information Science for Materials Discovery and Design*, pages 45–75. Springer, 2016.

[16] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, pages 937–945, 2014.

[17] M. Gelbart, J. Snoek, and R. Adams. Bayesian optimization with unknown constraints. In *ICML*, pages 250–259, Corvallis, Oregon, 2014. AUAI Press.

[18] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[19] J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence. Batch bayesian optimization via local penalization. In *AISTATS*, pages 648–657, 2016.

[20] N. gov. NYC Trip Record Data. http://www.nyc.gov/html/tlc/, 2016. Last accessed on 2016-10-10.

[21] J. Harold, G. Kushner, and G. Yin. *Stochastic approximation and recursive algorithm and applications*. Springer, 2003.

[22] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012.

[23] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pages 918–926, 2014.

[24] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 34(3):441–466, 2006.

[25] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[26] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[27] A. Jameson. Re-engineering the design process through computation. *Journal of Aircraft*, 36(1):36–50, 1999.

[28] R. P. A. e. Jasper Snoek, Hugo Larochelle. Spearmint. https://github.com/HIPS/Spearmint, 2015.

[29] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[30] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[31] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Poczos. Multi-fidelity gaussian process bandit optimisation. *arXiv preprint arXiv:1603.06288*, 2016.

[32] K. Kandasamy, G. Dasarathy, J. Schneider, and B. Poczos. Multi-fidelity bayesian optimisation with continuous approximations. *arXiv preprint arXiv:1703.06240*, 2017.

[33] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pages 295–304, 2015.

[34] T. Kathuria, A. Deshpande, and P. Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

[35] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016.

[36] O.-P. Koistinen, E. Maras, A. Vehtari, and H. Jónsson. Minimum energy path calculations with gaussian process regression. *Nanosystems: Physics, Chemistry, Mathematics*, 7(6), 2016.

[37] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 86(1):97–106, 1964.

[38] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[39] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[40] P. L'Ecuyer. A unified view of the IPA, SF, and LR gradient estimation techniques. *Management Science*, 36(11):1364–1383, 1990.

[41] P. L'Ecuyer. Note: On the interchange of derivative and expectation for likelihood ratio derivative estimators. *Management Science*, 41(4):738–747, 1995.

[42] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

[43] D. J. Lizotte. *Practical bayesian optimization.* PhD thesis, University of Alberta, 2008.

[44] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[45] D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

[46] A. Mahajan and D. Teneketzis. Multi-armed bandit problems. In D. C. A. O. Hero III, D. A. Castanon and K. Kastella, editors, *Foundations and Applications of Sensor Management*. Springer-Verlag, 2007.

[47] N. Mahendran, Z. Wang, F. Hamze, and N. De Freitas. Adaptive mcmc with bayesian optimization. In *Artificial Intelligence and Statistics*, pages 751–760, 2012.

[48] S. Marmin, C. Chevalier, and D. Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. In *International Workshop on Machine Learning, Optimization and Big Data*, pages 37–48. Springer, 2015.

[49] S. Marmin, C. Chevalier, and D. Ginsbourger. Efficient batch-sequential bayesian optimization with moments of truncated gaussian vectors. *arXiv preprint arXiv:1609.02700*, 2016.

[50] P. Milgrom and I. Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

[51] J. Mockus. *The bayesian approach to local optimization*. Springer, 1989.

[52] J. Mockus. *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media, 2012.

[53] J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[54] I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of la-

tent gaussian models. In *Advances in Neural Information Processing Systems*, pages 1732–1740, 2010.

[55] D. M. Negoescu, P. I. Frazier, and W. B. Powell. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[56] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In *3rd international conference on learning and intelligent optimization (LION3)*, pages 1–15. Citeseer, 2009.

[57] F. Pedregosa. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, pages 737–746, 2016.

[58] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 55(1):2–13, 2013.

[59] R.-É. Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 2006.

[60] M. Poloczek, J. Wang, and P. I. Frazier. Multi-information source optimization. arXiv preprint arXiv:1603.00389, 2016.

[61] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[62] W. Scott, P. Frazier, and W. Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026, 2011.

[63] A. Shah and Z. Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pages 3312–3320, 2015.

[64] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[65] S. P. Smith. Differentiation of the cholesky algorithm. *Journal of Computational and Graphical Statistics*, 4(2):134 – 147, 1995.

[66] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[67] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. In *ICML*, pages 2171–2180, 2015.

[68] J. Snoek, K. Swersky, R. Zemel, and R. Adams. Input warping for bayesian optimization of non-stationary functions. In *ICML*, pages 1674–1682, 2014.

[69] J. Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer Science & Business Media, 2005.

[70] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, pages 4134–4142, 2016.

[71] N. Srinivas, A. Krause, M. Seeger, and S. M. Kakade. Gaussian process

optimization in the bandit setting: No regret and experimental design. In *ICML*, pages 1015–1022, 2010.

[72] K. Swersky, J. Snoek, and R. P. Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[73] B. S. Thomson, J. B. Bruckner, and A. M. Bruckner. *Elementary real analysis*. ClassicalRealAnalysis. com, 2008.

[74] A. Törn and A. Zilinskas. Global optimization, volume 350 of lecture notes in computer science, 1989.

[75] J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Metrics optimization engine. http://yelp.github.io/MOE/, 2014. Last accessed on 2016-01-21.

[76] J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Parallel bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016.

[77] Y. Wang, K. G. Reyes, K. A. Brown, C. A. Mirkin, and W. B. Powell. Nested-batch-mode learning and stochastic optimization with an application to sequential multistage testing in materials science. *SIAM Journal on Scientific Computing*, 37(3):B361–B381, 2015.

[78] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Ensemble bayesian optimization. *arXiv preprint arXiv:1706.01445*, 2017.

[79] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, N. De Freitas, et al. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013.

[80] A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, pages 1067–1075, 2013.

[81] A. G. Wilson, C. Dann, and H. Nickisch. Thoughts on massively scalable gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.

[82] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016.

[83] A. G. Wilson and H. Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *ICML*, pages 1775–1784, 2015.

[84] J. Wu and P. Frazier. The parallel knowledge gradient method for batch bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 3126–3134, 2016.

[85] J. Wu and P. I. Frazier. Discretization-free knowledge gradient methods for bayesian optimization. *arXiv preprint arXiv:1707.06541*, 2017.

[86] J. Wu and Q. Huang. Graphene growth process modeling: a physical–statistical approach. *Applied Physics A*, 116(4):1747–1756, 2014.

[87] J. Wu, M. Poloczek, A. G. Wilson, and P. I. Frazier. Bayesian optimization with gradients. *arXiv preprint arXiv:1703.04389*, 2017.